Nr. 1/86 Januar
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

peeker

Quicksort
Vokabeltrainer
ProDOS-Konverter
Zeichengenerator
Kyan-Pascal
Video-Karte

Testbericht Kyan-Pascal

Hüthig PUBLIKATION

APPLEWORKS

SYSTEMAUFBAU · DATENBANK
SCHREIBTISCHMANAGER · RECHENBLATT

V. BOTTA / CHR. LANGE / K. ZIMMERMANN

te-wi

APPLEWORKS

TEXTBEARBEITUNG · ACCESS II · DATENFERNÜBERTRAGUNG · SYSTEMINFORMATIONEN

V. BOTTA / CHR. LANGE / K. ZIMMERMANN

te-wi

Band 1:
1. Einleitung
2. Was Sie benötigen
3. Starten von APPLE WORKS
4. Der Schreibtischmanager
5. Datenbank
6. Rechenblatt

A1 Anschluß der Festplatte ProFile
A2 APPLE II Easy Pieces Referenz
A3 Druckeranpassungen
A4 DOS 3.3 Konvertierungen
A5 APPLE WORKS Disketten sichern/kopieren
A6 Hilfsfunktionen nach Programmteilen

Band 2:
1. Einleitung
2. Was Sie benötigen
3. Starten von APPLE WORKS
4. Der Schreibtischmanager
5. Textbearbeitung
6. Datenfernübertragung

A1 ... A6 wie Band 1, dazu:
A7 Modemkabel für APPLE IIe, IIc
A8 SuperSerialCard Einstellung
A9 ASCII-Textdateien aus anderen Dateiformaten für ACCESS II
A10 DATEX-P20 F Verzeichnis
A11 Deutsche/Englische Menübilder von ACCESS II

APPLE WORKS auf APPLE II, IIe, IIc:
verwandelt APPLE-II-Computer in einen Elektronischen Schreibtischmanager mit:
Texterstellung ... Edition, Briefarchiv, Ausdruck etc. Datenarchivierung ... Kontoführung, Buchhaltung etc. Formblattkalkulation ... Bilanzen, VisiCalc-Dateien etc. Datenfernübertragung ... Mailbox, Rechnerkopplung etc.

- ist ein erfolgreicheres Integrationspaket als LOTUS auf IBM PC!
- ist auf 1 MByte Speichererweiterungen Ihres APPLE II vorbereitet!
- erschließt Ihnen die Zukunftstechnik MAILBOX!
- ist ebenso einfach zu bedienen wie APPLE WRITER:
  Kein Befehlsstudium ... Einfachste Menüführung ... Sofortige Anwendbarkeit

te-wi's APPLE WORKS SYSTEMBÜCHER 1+2 zeigen Ihnen:
- Sämtliche APPLE WORKS Funktionen an Beispielen aus der Wirtschaft
- Das Wechseln zwischen Text/Rechenblatt/Datenarchiv/Dfü
- Umfassende Systeminformationen zu Dateikonvertierung, Druckeranpassung etc.

Von Botta/Lange/Zimmermann je 264 Seiten, Softcover, je DM 49,–

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

te-wi

Weitere te-wi-Bücher

NEU

Das APPLE II/II+/IIe/IIc-Handbuch
(L. Poole)
Erst mit Hilfe dieses Leitfadens werden Sie Ihren Apple II erfolgreich einsetzen, denn Text und Bildmaterial gehen weit über das hinaus, was herstellerseitig an Literatur angeboten wird.
Neu überarbeitet und jetzt um die spezifischen Eigenheiten der Modelle **II e** und **II c** erweitert. 472 Seiten, Softcover, DM 66,–

LOGO – Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den USA. Für die Computer APPLE II, C-64, IBM PC, ATARI bis 520 ST., TI-99 und Schneider CPCs.
Hochwertiges Textbuch für Logo-Kurse zu Hause und im Lehrbereich.
384 Seiten, A4, DM 59,–

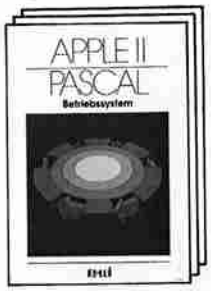
NEU

APPLE II – Bewegte 3D-Graphik
(Phil Cohen)
Selbstentworfene Graphiken und Diagramme – animiert oder als Standbilder – eben oder räumlich: alle erforderlichen BASIC-Programme mit Erklärung finden Sie in diesem Buch.
200 Seiten, Softcover, DM 49,–

Apple Maschinensprache
Für BASIC-Programmierer der einfachste Zugang zur Muttersprache des APPLE. Wesentlich schnellere Maschinenprogramme, direkte Manipulation des Mikroprozessors 6502 im APPLE – als Brücke dorthin benötigt dieses Buch nur die drei BASIC-Befehle POKE, CALL, PEEK. D. Inman/K. Inman, DM 49,–

NEU

Reparaturanleitung Computer: Apple II, IIplus
Einzigartige Serviceunterlage für Reparaturen und Entwicklungsarbeiten am Apple II. Enthält Schaltpläne, Bauteile- und Vergleichstypenliste; Prüfpunkte mit Oszillogrammen der Signalformen, Logiktabellen, Spannungsangaben; schnelle Servicetests; Anleitung zur systematischen Fehlersuche.
In A4-Mappe, DM 29.80



Erstes deutsches Referenzwerk sämtlicher Befehle und Systemroutinen von Apple II, IIplus, IIe
APPLE II PASCAL NEU
Betriebssystem, 272 S., DM 49,–
Sprache, 216 S., DM 39,–
Pascal 1.2 Addendum, 112 S., DM 36,–

Grundlagenbuch, Bestseller APPLE II PASCAL.
Eine praktische Anleitung.
544 S., DM 59,–

tm 4422

Noch im Programm:
Computer für Kinder, APPLE II, DM 29,80
6502 Programmieren in Assembler, DM 59,–
Umweltdynamik (Prospekt anfordern), DM 66,– (NEU)

Macintosh Programmierhandbuch mit MSBASIC 2.0 (Ende '85), DM 59,–
Einführung in die Mikrocomputer-Technik, DM 66,–
M68000-Familie, 2 Bände, DM 79,– und DM 69,–

60 JAHRE FACHLITERATUR

Über 40 erfolgreiche Fachzeitschriften und ein großes Fachbuchprogramm für Wirtschaft, Wissenschaft und Technik erscheinen bei der Verlagsgruppe Dr. Alfred Hüthig. Sie sind unerläßlich zur aktuellen Information und beruflichen Weiterbildung der Fachleute in den Bereichen:
Aerosole · Chemie · Makromolekulare Chemie · Chromatographie · Computeranwendung im Labor · Metall · Gummi · Kautschuk · Kunststoffe · Kosmetika · Parfümerie · Elektronik · Elektrotechnik · CAD, CAM · Elektrohandwerk · Elektrohandel · Energietechnik · Beleuchtungstechnik · Hobbyelektronik · EDV · Mikrocomputer · Fernsehen · Kinotechnik · Nachrichtenelektronik · Radio- und Fernsehtechnik · Unterhaltungselektronik · Kriminalistik · Sicherungstechnik · Recht · Verwaltung · Bundesbahn · Kunsthandel · Schuhtechnik · Verpackung und Zahnmedizin.

Hüthig
1925 – 1985

Die Peeker-Redaktion befindet sich übrigens in der zweiten Etage ganz links

inhalt

Impressum
Peeker
3. Jahrgang 1986
ISSN 0176–9200
© für den gesamten Inhalt einschließlich der Programme Dr. Alfred Hüthig Verlag, Heidelberg 1986

Verleger und Herausgeber:
Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften:
Heinz Melcher

Chefredakteur: Ulrich Stiehl (us)

Redaktion: Harald Grumser

Sekretariat, Tel. (0 62 21) 48 92 31
Bestellungen, Tel. (0 62 21) 48 92 78

Anzeigenleitung:
Jürgen Maurer, Tel. (0 62 21) 48 92 18
z. Zt. gilt Anzeigenpreisliste Nr. 4
Vertriebsleitung:
Walter Menzel, Tel. (0 62 21) 48 92 80
Produktionsleitung: Gunter Sokollek
Gestaltung: Rainer Schmitt
Illustrationen. Hannes Pohle

Hüthig
PUBLIKATION

peeker

Heft 1/1986

Grundlagen

Wie funktioniert Quicksort?
Mit einem Demonstrationsprogramm
von Ulrich Stiehl 6

Utility

Quicksort
Eine superschnelle Applesoft-Erweiterung
von Harald Grumser 16

Schule

Vokabeltrainer
Vokabeln lernen mit dem Apple II
von Thomas Zink 20

Grafik

Applesoft-HGR-Erweiterung
in Assembler
von Klaus Schäfer 30

ProDOS

PROTODOS
Konvertierung von ProDOS-
in DOS-3.3-Dateien
von Arne Schäpers 36

UCSD

Designer
Zeichensatz-Generator
in Apple-Pascal
von Gabor Herr 43

Turbo

READPAS
Konvertierung von UCSD-
in Turbo-Pascal-Textfiles
von Michael Erperstorfer 48

Kyan

Kyan-Pascal
Ein vollcompilierendes
6502-Pascal
Erfahrungsbericht
von Herbert Pohl 50

Praxis

Der Apple II als Werkzeug
für den Betriebswirt
Teil 1: Das Simplex-Verfahren
in der Controller-Praxis
von Dipl.-Betriebswirt Willy Holtkamp 56

Hobby

Grafik-Demonstrationen
Teil 2
von Ralf Knoke 61

Testberichte

Video-Clips für jedermann
Die Video-1000-Karte
Erfahrungsbericht
von Norbert Man Brandl 63

The Write Choice
Ein preiswertes Textverarbeitungsprogramm
Erfahrungsbericht
von Franz-Josef Hüskens 66

Hard- und Software zum Apple IIc
zusammengestellt von Matthias Pohl 67

Ampersoft
Eine Ampersand-Utility-Sammlung
Kurzbericht
von Franz-Josef Hüskens 68

Buchbesprechungen 69

Inserentenverzeichnis 70

Verlag:
Dr. Alfred Hüthig Verlag GmbH
Im Weiher 10, Postfach 10 28 69
6900 Heidelberg
Telefon (0 62 21) 4 89-1
Telex 4-6 17 27 hued d.
Telefax (0 62 21) 489 279
BTX * 51851 #

Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement DM 72,–, einschließlich MwSt,
im Inland portofrei. Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089/319 10 67, Telex 0 522 656

Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postgirokonten: BRD: Karlsruhe 485 45-753; Österreich: Wien 755588; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 47718; Belgien: Brüssel 723026; Dänemark: Kopenhagen 349 69; Norwegen: Oslo 994 24; Schweden: Stockholm 5477 76-5

Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 0 2165 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Wie funktioniert Quicksort?

Mit einem Demonstrationsprogramm

von Ulrich Stiehl

Anfang der sechziger Jahre entwickelte ein englischer Informatiker namens C.A.R. Hoare ein neuartiges Sortierverfahren, das 1962 in einer Computerfachzeitschrift unter der Bezeichnung „Quicksort" vorgestellt wurde. Es sollte sich als der bislang beste Sortieralgorithmus erweisen. Wie alle genialen Algorithmen scheint auch Quicksort dem Lernenden zunächst unüberwindlich kompliziert zu sein. Hat man den Algorithmus jedoch erst einmal verstanden, dann enthüllt er sich als verblüffend einfach.

1. Sortieren und Vergleichen

Der Begriff des Sortierens ist nicht so einfach, wie man gemeinhin denkt.

Alphabetische Sortierung

Betrachten wir zunächst das Alphabet. Liegen beispielsweise die Buchstaben des Abc sortiert vor? „A" ist ein Vokal, „B" ein Labial, „C" ein kombinierter Konsonant („T" + „Z"), „D" ein Dental, „E" wieder ein Vokal usw. Von einer Sortierung kann keine Rede sein! Wer zufällig einmal ein Wörterbuch der altindischen Sprache Sanskrit in die Hand bekommen sollte, wird zu seinem Erstaunen feststellen, daß die alten Inder das richtig sortierte Alphabet erfunden haben. Zunächst kommen die Vokale und dann die Konsonanten, die jeweils in sich exakt teilsortiert sind. Die Dentale sind beispielsweise in der Reihenfolge „T, TH, D, DH, N" angeordnet. „T" ist der stimmlose Dental, „TH" der stimmlos aspirierte Dental, „D" der stimmhafte Dental, „DH" der stimmhaft aspirierte Dental, und schließlich ist „N", wofür es ein spezielles Zeichen gibt, dasjenige Dental-N, das einem anderen Dental vorausgeht. Das „N" in „Ente" spricht man nämlich anders als das „N" in „Anker" aus, denn bei „En*te*" steht ein Dental-N vor dem Dental-T und bei „An*k*er" ein Guttural-N vor dem Guttural-K. Eines können wir auf jeden Fall festhalten: Wer auch immer das lateinische Alphabet erfunden hat, muß eine Vorliebe für „chaotisches Sortieren" gehabt haben.

Numerische Sortierung

Auch die Sortierung von Zahlen ist nicht so eindeutig, wie man zunächst annehmen möchte. Die Folge
1, 2, 3
wird man gemeinhin als aufsteigend sortiert bezeichnen. Wie steht es hingegen mit der Folge
739, 1341, 522, 1862,
die in einem Intelligenztest-Buch von H.J.Eysenck steht? Ist sie nach einem „geheimen Verfahren" sortiert? Tatsächlich gibt es in der höheren Mathematik „sortierte" Reihen, die man landläufig weder als aufsteigend noch absteigend bezeichnen würde.

Sonstige Sortierung

Noch undurchsichtiger wird der Begriff des Sortierens, wenn weder Wörter noch Zahlen sortiert werden sollen. Wie kann man beispielsweise die Shapes eines Grafikspiels sortieren: nach der Anzahl der Bits, nach der Richtung der Vektoren, nach der Wirkung auf den Spieler? Hier wie auch bei den Wörtern und Zahlen müssen jeweils spezielle „Größer-Kleiner-Gleich"-**Vergleichskriterien** festgelegt werden. Für die alphabetische Sortierung „nach Duden" gilt u.a. (Beispiele):
„A" < „B"
„a" = „A"
„ä" = „A"
„ß" = „SS"
„-" = nichts
„AA" < „AB"
usw.
Dem Sortieren liegt also ein Vergleichsprozeß zugrunde, der von den Vergleichskriterien bestimmt wird. Ob eine Sortierung richtig oder falsch ist, hängt damit allein von den Vergleichskriterien ab, die ihrerseits völlig willkürlich festgelegt sein können, wie das lateinische Alphabet zeigt.

2. Sortieren und Tauschen

Unter Sortieren verstehen wir die Neuanordnung einer bestimmten Anzahl von Elementen nach zuvor festgelegten Vergleichskriterien. Programmtechnisch werden praktisch immer **Arrays** (= dimensionierte oder Feldvariablen) sortiert, und zwar bei Mikrocomputern üblicherweise im internen Speicher (= **internes Sortieren** im RAM) und nur ausnahmsweise im externen Speicher (= **externes Sortieren** direkt auf dem Datenträger = Diskette usw.). Betrachten wir hierzu das folgende Applesoft-Programm-Fragment:
10 W (1) = 25
20 W (2) = 30
30 W (3) = 10
Hier umfaßt der Array W nur 3 Elemente, nämlich
1. Element mit Wert 25,
2. Element mit Wert 30,
3. Element mit Wert 10.
Dabei ist zu unterscheiden zwischen der Nummer eines Elements (1., 2., 3.) und dem Wert eines Elements (25, 30, 10). Die Nummer eines Elements heißt auch Index und ist mathematisch gesehen eine Ordinalzahl = Ordnungszahl (Gegensatz: Kardinalzahl = Grundzahl).
Würden die 3 Werte 25, 30 und 10 als 3 entsprechend bezifferte Karteikärtchen vorliegen, so könnte man diese auf dem Tisch mit einem „flotten Griff" so verdrehen, daß sich sofort die aufsteigend sortierte Folge 10, 25, 30 ergeben würde.
Bei einem Sortierprogramm muß statt des „flotten Griffs" stets ein **paarweiser Tausch** vorgenommen werden, wobei in Applesoft in Ermangelung eines SWAP- oder direkten Tauschbefehls sogar ein indirekter Tausch über eine Zwischen- oder Dreieck-Tauschvariable T erfolgen muß.
Wie ersichtlich, sind bei unserem Array 2 Tauschoperationen erforderlich, um die gewünschte aufsteigende Sortierung herbeizuführen:

Vorher:
W (1) = 25
W (2) = 30
W (3) = 10
Erster Tausch von
1. und 3. Element:
T = W (1)
W (1) = W (3)
W (3) = T
Danach:
W (1) = 10
W (2) = 30
W (3) = 25
Zweiter Tausch von
2. und 3. Element:
T = W (2)
W (2) = W (3)
W (3) = T
Danach
W (1) = 10
W (2) = 25
W (3) = 30

Wert- und Indextausch

Da die Werte hier physisch vertauscht wurden, sprechen wir auch vom Werttausch. Das Gegenstück dazu ist der Indextausch. Da indes der Index oder die Feldnummer nur als „Offset", d.h. als relativer oder errechneter Abstand zum Beginn eines Arrays, und mithin gar nicht physisch existiert, muß zu diesem Zweck neben dem Wert-Array W ein Index-Array I eingeführt werden, der zunächst wie folgt initialisiert wird:
I (1) = 1
I (2) = 2
I (3) = 3
Dies besagt, daß der Index-Array I anfangs als Werte die Nummern des Wert-Arrays W übernimmt. Nach Beendigung der Sortierung gilt für den Index-Array:
I (1) = 3
I (2) = 1
I (3) = 2
Beispielsweise enthält das 1. Element des Index-Arrays I den Wert 3 und zeigt damit auf das 3. Element des Wert-Arrays W, das den Wert 10 enthält, so daß gilt:
W (I(1)) = W (3) = 10
Wenn man anstelle des Wert-Arrays den entsprechenden Index-Array sortiert, so bleibt die ursprüngliche Reihenfolge der Elemente des Wert-Arrays erhalten. Nehmen wir an, Sie hätten eine Kundendatei (Wert-Array), wobei jedem Kunden eine logische Suchnummer vergeben wurde, die der Nummer (= dem Index) des jeweiligen Datensatzes (= Wert-Elements) entspricht. Durch physische Vertauschung der Wert-Elemente würden die logischen Suchnummern verlorengehen. Deshalb

Tabellarisches Quicksort-Protokoll

01:	35	43	98	66	54	31	94	83	32	72	56	27	71	81	45	34	86	18	90	29	unsortiert
02:	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	Mitte 10,V:72
03:	**72**	43	98	66	54	31	94	83	32	**35**	56	27	71	81	45	34	86	18	90	29	Ersttausch
04:	72	43	**29**	66	54	31	94	83	32	35	56	27	71	81	45	34	86	18	90	**98**	Tausch
05:	72	43	29	66	54	31	**18**	83	32	35	56	27	71	81	45	34	86	**94**	90	98	Tausch
06:	72	43	29	66	54	31	18	**34**	32	35	56	27	71	81	45	**83**	86	94	90	98	Tausch
07:	72	43	29	66	54	31	18	34	32	35	56	27	71	**45**	**81**	83	86	94	90	98	Tausch
08:	**45**	43	29	66	54	31	18	34	32	35	56	27	71	**72**	81	83	86	94	90	98	Letzttausch
09:	**01**	02	03	04	05	06	07	08	09	10	11	12	**13**	14	15	16	17	18	19	20	Stack,F:15,L:20
10:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	**15**	**16**	**17**	**18**	**19**	**20**	Mitte 17,V:86
11:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	**86**	83	**81**	94	90	98	Ersttausch
12:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	**81**	83	**86**	94	90	98	Letzttausch
13:	**01**	02	03	04	05	06	07	08	09	10	11	12	**13**	14	15	16	17	**18**	19	**20**	Stack,F:15,L:16
14:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	**15**	**16**	17	18	19	20	Mitte 15,V:81
15:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	**81**	83	86	94	90	98	Ersttausch
16:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	**81**	83	86	94	90	98	Letzttausch
17:	**01**	02	03	04	05	06	07	08	09	10	11	12	**13**	14	15	16	17	18	19	20	Stack,F:18,L:20
18:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	**18**	**19**	**20**	Mitte 19,V:90
19:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	81	83	86	**90**	**94**	98	Ersttausch
20:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	81	83	86	**90**	94	98	Letzttausch
21:	**01**	02	03	04	05	06	07	08	09	10	11	12	**13**	14	15	16	17	18	19	20	Stack,F:19,L:20
22:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	**19**	**20**	Mitte 19,V:94
23:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	81	83	86	90	**94**	98	Ersttausch
24:	45	43	29	66	54	31	18	34	32	35	56	27	71	72	81	83	86	90	**94**	98	Letzttausch
25:	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	14	15	16	17	18	19	20	Mitte 7,V:18
26:	**18**	43	29	66	54	31	**45**	34	32	35	56	27	71	72	81	83	86	90	94	98	Ersttausch
27:	**18**	43	29	66	54	31	45	34	32	35	56	27	71	72	81	83	86	90	94	98	Letzttausch
28:	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	14	15	16	17	18	19	20	Mitte 7,V:45
29:	18	**45**	29	66	54	31	**43**	34	32	35	56	27	71	72	81	83	86	90	94	98	Ersttausch
30:	18	45	29	**27**	54	31	43	34	32	35	56	**66**	71	72	81	83	86	90	94	98	Tausch
31:	18	45	29	27	**35**	31	43	34	32	**54**	56	66	71	72	81	83	86	90	94	98	Tausch
32:	18	**32**	29	27	35	31	43	34	**45**	54	56	66	71	72	81	83	86	90	94	98	Letzttausch
33:	01	**02**	03	04	05	06	07	**08**	09	10	11	12	13	14	15	16	17	18	19	20	Stack,F:10,L:13
34:	01	02	03	04	05	06	07	08	09	**10**	**11**	**12**	**13**	14	15	16	17	18	19	20	Mitte 11,V:56
35:	18	32	29	27	35	31	43	34	45	**56**	**54**	66	71	72	81	83	86	90	94	98	Ersttausch
36:	18	32	29	27	35	31	43	34	45	**54**	**56**	66	71	72	81	83	86	90	94	98	Letzttausch
37:	01	**02**	03	04	05	06	07	**08**	09	10	11	12	13	14	15	16	17	18	19	20	Stack,F:12,L:13
38:	01	02	03	04	05	06	07	08	09	10	11	**12**	**13**	14	15	16	17	18	19	20	Mitte 12,V:66
39:	18	32	29	27	35	31	43	34	45	54	56	**66**	71	72	81	3	86	90	94	98	Ersttausch
40:	18	32	29	27	35	31	43	34	45	54	56	**66**	71	72	81	83	86	90	94	98	Letzttausch
41:	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	09	10	11	12	13	14	15	16	17	18	19	20	Mitte 5,V:35
42:	18	**35**	29	27	**32**	31	43	34	45	54	56	66	71	72	81	83	86	90	94	98	Ersttausch
43:	18	35	29	27	32	31	**34**	**43**	45	54	56	66	71	72	81	83	86	90	94	98	Tausch
44:	18	**34**	29	27	32	31	**35**	43	45	54	56	66	71	72	81	83	86	90	94	98	Letzttausch
45:	01	**02**	**03**	**04**	**05**	**06**	07	08	09	10	11	12	13	14	15	16	17	18	19	20	Mitte 4,V:27
46:	18	**27**	29	**34**	32	31	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Ersttausch
47:	18	**27**	29	34	32	31	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Letzttausch
48:	01	02	**03**	**04**	**05**	**06**	07	08	09	10	11	12	13	14	15	16	17	18	19	20	Mitte 4,V:34
49:	18	27	**34**	**29**	32	31	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Ersttausch
50:	18	27	**31**	29	32	**34**	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Letzttausch
51:	01	02	**03**	**04**	**05**	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	Mitte 4,V:29
52:	18	27	**29**	**31**	32	34	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Ersttausch
53:	18	27	**29**	31	32	34	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Letzttausch
54:	01	02	03	**04**	**05**	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	Mitte 4,V:31
55:	18	27	29	**31**	32	34	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Ersttausch
56:	18	27	29	**31**	32	34	35	43	45	54	56	66	71	72	81	83	86	90	94	98	Letzttausch
57:	18	27	29	31	32	34	35	43	45	54	56	66	71	72	81	83	86	90	94	98	sortiert

Leser werben Leser

»peeker« bietet Ihnen was!

Wer jetzt schenkt hat mehr von seinem Apple. Dafür schenkt »peeker« Ihnen etwas: Den praktischen Disk-Locher, der die Speicherkapazität Ihrer Disketten verdoppelt!

Sie wissen ja, wie gut der »peeker« Ihnen im täglichen Umgang mit Ihrem Apple behilflich ist. Und Sie brauchen Ihren »peeker« nicht mehr zu teilen oder auszuleihen.

Der blaue Disketten-Locher ist unser Geschenk an Sie für einen neuen »peeker«-Abonnenten. Denn wer einen Apple hat, der soll auch seinen »peeker« haben.

Bestellcoupon

Ich habe den neuen Abonnenten geworben und erhalte kostenlos den Disk-Locher.

Name, Vorname

Straße, Postfach

PLZ, Ort

X

Datum, Unterschrift

Ich bin der neue Abonnent. Bitte liefern Sie mir bis auf Widerruf, zumindest aber für 1 Jahr, »peeker« zum Jahresbezugspreis von DM 72,– (Ausland plus DM 18,– Porto) an folgende Anschrift:

Name, Vorname

Straße, Postfach

PLZ, Ort

X

Datum, Unterschrift

Gewünschte Zahlungsweise
☐ gegen Rechnung
☐ bargeldlos durch Bankeinzug

Konto-Nr. Bankleitzahl

Geldinstitut

Vertrauensgarantie:
Diese Bestellung kann ich innerhalb einer Woche bei Dr. Alfred Hüthig Verlag GmbH, Im Weiher 10, 6900 Heidelberg 1 widerrufen. Zur Wahrung der Frist genügt die rechtzeitige Absendung. Ich bestätige die Kenntnisnahme mit meiner Unterschrift:

X

2. Unterschrift

Peeker 1/86

Coupon ausschneiden und einsenden an:

»peeker«
Abonnementservice
Im Weiher 10
6900 Heidelberg 1

wären hier die Indizes (= Zeiger) und nicht die Werte zu sortieren.

3. Sortieren mit Bubblesort

Bei der Folge
1. 25
2. 30
3. 10
haben wir „auf den ersten Blick" erkannt, daß 10 die kleinste Zahl ist; folglich haben wir 10 mit 25 vertauscht, damit 10 zum kleinsten Element wurde. Danach haben wir bei
1. 10
2. 30
3. 25
„auf den zweiten Blick" erkannt, daß 25 die zweitkleinste Zahl ist; folglich haben wir 30 mit 25 vertauscht, womit 30 zum drittkleinsten (= größten) Element wurde.
Dieses „intuitive Sortierverfahren" ist nicht nur für den Computer, sondern – bei größeren Mengen von zu sortierenden Elementen – auch für den Menschen völlig ungeeignet. Sie glauben mir nicht? Dann unterbrechen Sie bitte die Lektüre dieses Aufsatzes und suchen Sie einmal, nachdem Sie sich das heutige Datum notiert und vorsorglich eine Krankmeldung an Ihren Arbeitgeber geschickt haben, das fünfzigkleinste Wort aus diesem Peeker-Heft heraus. ... Wieviele Tage sind seit Beginn der „Suchaktion" verstrichen?

Holzhammer und Köpfchen

Sieht man einmal von der intuitiven Methode ab, so lassen sich die Sortierverfahren in zwei Gruppen einteilen:
a) Holzhammer-Algorithmus
b) Köpfchen-Algorithmus.

Da Programme in Assembler bis zu hundertmal schneller als in BASIC oder Pascal sind, bedient man sich in anderen Bereichen häufig primitiver, aber sehr schneller Assembler-Programme. Bei Sortierprogrammen versagen jedoch die Holzhammer-Verfahren, denn es läßt sich leicht nachweisen, daß ein Quicksort-Programm in Applesoft schneller als ein Bubblesort-Programm in Assembler ist, z.B. wenn ein Array mit 1000 Zahlen sortiert werden soll. Betrachten wir zu diesem Zweck den Bubblesort oder die „Blasensortierung" als das primitivste Holzhammer-Verfahren. Der Bubblesort durchläuft wiederholt alle Elemente vom ersten bis zum vorletzten und vergleicht dabei jeweils das nächste Elements (X + 1) mit dem momentanen Element (X). Wenn das nächste Element kleiner als das momentane ist, werden beide Elemente vertauscht. Die Sortierung wird beendet, wenn bei einem (= dem letzten) Durchlauf keine Vertauschung mehr erforderlich war. Beispiel anhand unserer 3 Zahlen:

Erster Durchlauf
1. 25
2. 30
3. 10
2. mit 1. Element vergleichen (V1)
30 < 25: nein, kein Tausch
3. mit 2. Element vergleichen (V2)
10 < 30: ja, tauschen (T1)
Es wurde getauscht, also weiter.

Zweiter Durchlauf
1. 25
2. 10
3. 30
2. mit 1. Element vergleichen (V3)
10 < 25: ja, tauschen (T2)
3. mit 2. Element vergleichen (V4)
30 < 25: nein, kein Tausch
Es wurde getauscht, also weiter.

Dritter Durchlauf
1. 10
2. 25
3. 30
2. mit 1. Element vergleichen (V5)
25 < 10: nein, kein Tausch
3. mit 2. Element vergleichen (V6)
30 < 25: nein, kein Tausch
Es wurde nicht getauscht, also Ende.

Insgesamt waren hier nur 2 Tauschoperationen (T2) und 6 Vergleichsoperationen (V6) erforderlich. In Wirklichkeit ist der Bubblesort, bei dem mit jedem Durchlauf die nächstgrößere Zahl wie eine Blase (= bubble) im Wasser nach oben transportiert wird, das ineffizienteste Sortierverfahren überhaupt.
Die Effizienz eines Sortierverfahrens hängt von der Anzahl der Tausch- und Vergleichsoperationen ab. Als Faustregel können Sie davon ausgehen, daß Sie für das Sortieren eines Arrays mit 1000 Elementen beim Bubblesort mit ca. 1.000.000 Vergleichen und bei Quicksort mit 10.000 Vergleichen rechnen müssen. Damit ist der Quicksort – wiederum als Faustregel – etwa 100mal schneller als der Bubblesort. Daraus erhellt zugleich, daß ein Assembler-Bubblesort kaum eine Chance gegen einen Applesoft-Quicksort hat.

4. Sortieren mit Quicksort

Vorsortierung

Angenommen, Sie hätten die unangenehme Pflicht, 10.000 Adreß-Karteikärtchen manuell nach dem Alphabet zu sortieren. Dann würden Sie sicherlich zunächst eine **Vorsortierung** vornehmen, indem Sie getrennte „Häufchen" mit den „A"-Adressen, „B"-Adressen usw. anlegen würden. Das sich dann beispielsweise ergebende „S"-Häufchen würden Sie ferner in „ST"- und „SCH"-Teilhäufchen zerlegen usw. Jedes der Häufchen würde einen **Bereich** (= Abschnitt) darstellen, der insgesamt alphabetisch kleiner oder größer als die anderen Häufchen ist, d.h. alle Kärtchen des „B"-Häufchens sind z.B. kleiner als die Kärtchen des „C"-Häufchens, aber größer als die Kärtchen des „A"-Häufchens. Doch sind die Kärtchen des „B"-Häufchens zunächst noch nicht *in sich* sortiert.

Mechanische Zerlegung

Quicksort arbeitet nach dem soeben skizzierten Prinzip der Aufteilung oder Zerlegung in Bereiche (englisch „partition"), doch werden die Bereiche nicht nach sachlichen Kriterien, sondern rein mechanisch gebildet. Bei alphabetischer Sortierung sind die Buchstaben-Häufchen sachlich begründbar, obwohl Sie sehr schnell merken werden, daß manche Anfangsbuchstaben extrem häufig (z.B. „S") und andere extrem selten (z.B. „Y") vorkommen. Wenn Sie jedoch 10.000 Zahlen, deren „Streuung" Sie nicht kennen, manuell zu sortieren hätten, dann würden Sie feststellen, daß man bei größeren Datenmassen nur mit einer mechanischen Aufteilung weiterkommt, zumal ein Computer mit „sachlichen Kriterien" ohnehin nichts anfangen kann.

Quicksort-Protokoll

Betrachten wir nunmehr die Tabelle, die die aufsteigende Sortierung von 20 Zufallszahlen im Werttausch-Verfahren Schritt für Schritt nachvollzieht. In der linken Spalte befinden sich die Zeilennummern
01:
02:
usw. bis
57:
Zeile 1 enthält 20 unsortierte Zahlen
01: 35, 43, 98...,
während Zeile 2 die Nummern (= Indizes) der Zahlen
02: 01, 02, 03...
angibt.

Bis zur ersten Teilung

Betrachten wir nun die Zeile 1 mit den 20 Ausgangswerten. Zunächst ist dieser Bereich 1-20 noch ungeteilt, wobei das 1. Element das linke oder untere Begrenzungselement und das 20. Element das rechte oder obere Begrenzungselement darstellt. Wie gelangen wir nun zur ersten

Teilung? Indem man eine willkürliche Halbierung vornimmt. Dies löst jedoch noch nicht das Problem des **Aufteilungskriteriums**. Quicksort vollzieht daher folgenden mechanischen Dreierschritt:

a) Zuerst wird ein sich etwa in der Mitte vom linken und rechten Begrenzungselement befindliches sog. **mittleres Element** selektiert, und zwar hier das 10. Element (s. Zeile 2, rechter Rand). *Man beachte, daß dieses mittlere Element später nicht notwendigerweise die Teilungsgrenze für die zwei Hälften darstellen wird.*

b) Dann wird der Wert dieses mittleren Elements (hier 72, s. Zeile 2, rechter Rand) willkürlich zum **Vergleichswert** erklärt.

c) Schließlich wird der Wert des mittleren Elements mit dem Wert des linken, untersten Elements vertauscht. Konkret heißt dies hier, daß der Wert 72 des 10. Elements mit dem Wert 35 des 1. Elements ausgetauscht wird (s. Zeile 3). Der Sinn dieses sog. **Ersttausches** hängt mit dem „Durchdrehen" des Algorithmus zusammen (s.u.).

Es ergibt sich damit in Zeile 3 folgende neue Ausgangssituation: Ein Vergleichswert (72) ist aufgrund des mittleren (10.) Elements bestimmt worden, und das unterste (1.) Element ist mit dem mittleren (10.) Element vertauscht worden. Wir betrachten weiterhin die Zeile 3:

Aufsteigendes Suchen: Da das unterste Element bereits durch den Ersttausch vorläufig „abhackt" worden ist, suchen wir ab dem zweituntersten (hier 2.) Element *von links nach rechts* einen Wert, der dem Vergleichswert *größer/gleich* ist. Die 43 als Wert des 2. Elements ist nicht größer als der Vergleichswert 72, wohl aber die 98 als Wert des 3. Elements.

Absteigendes Suchen: Nachdem wir die 98 gefunden haben, suchen wird ab dem obersten (hier 20.) Element *von rechts nach links* einen Wert, der dem Vergleichswert *kleiner/gleich* ist. Dies ist hier gleich bei dem 20. Element der Fall, dessen Wert 29 kleiner als der Vergleichswert 72 ist.

Normaler Tausch: Nachdem wir aufsteigend bzw. absteigend jeweils einen Wert gefunden haben, der größer/gleich bzw. kleiner/gleich in bezug auf den Vergleichswert ist, vertauschen wird diese gefundenen Werte. Das heißt hier konkret, daß die 98 mit der 29 vertauscht wird. Die erfolgte Vertauschung ist aus der Zeile 4 ersichtlich.

Weitere Suche: Da die 98 das 3. Element und die 29 das 20. Element waren, setzen wir unsere aufsteigende Suche mit dem 4. Element und die absteigende Suche mit dem 19. Element fort. Wir finden dann die Werte 94 (7. Element) und 18 (18. Element), deren erfolgte Vertauschung aus der Zeile 5 hervorgeht. Die Zeile 6 belegt noch die Vertauschung von 83 (8. Element) und 34 (16. Element) sowie die Zeile 7 die Vertauschung von 81 (14. Element) und 45 (15. Element).

Letzttausch: Wenn man jetzt, nachdem man von links beim 14. Element und von rechts beim 15. Element angelangt ist, die Suche fortsetzt, überkreuzen sich die Nummern der Elemente. Deshalb beenden wir mit der ersten Überkreuzung (Überlappung) das Suchen und vertauschen nunmehr den Wert des 14. Elements (45), der sich beim absteigenden Suchen ergab, mit dem Wert des 1. Elements (72), das vom Ersttausch herrührte und gleichzeitig während des gesamten Suchvorgangs den Vergleichswert darstellte. Wenn nämlich der beim überlappend absteigenden Suchen gefundene Wert 45 kleiner als der Vergleichswert 72 ist und das 1. Element den Vergleichswert beinhaltet, dann können wir getrost 45 mit 72 vertauschen, womit der Vergleichswert 72 als 14. Element seine **endgültige Position** erhält. Das Ergebnis des Letzttausches ist aus Zeile 8 ersichtlich.

Was haben wir damit erreicht?
1. Der Bereich der Elemente 1-13 enthält jetzt nur noch Werte, die dem Wert des 14. Elements, d.h. dem Vergleichswert, kleiner oder gleich sind.
2. Der Bereich der Elemente 15-20 enthält jetzt nur noch Werte, die dem Wert des 14. Elements, d.h. dem Vergleichswert, größer oder gleich sind.
3. Der Wert des 14. Elements, d.h. der Vergleichswert, ist bereits endgültig, d.h. die Position des 14. Elements wird sich bei den nachfolgenden Teilungen und Suchvorgängen nicht mehr ändern. Damit haben wird zwei vorsortierte Teilbereiche gewonnen, nämlich den Bereich 1-13 sowie den Bereich 15-20, denn der Bereich 1-13 ist in bezug auf den Bereich 15-20 *insgesamt* kleiner oder gleich. Die zwei Teilbereiche sind jedoch noch nicht *in sich* sortiert.
4. Man beachte, daß die Teilung nicht direkt mit dem mittleren Element zusammenhängt. Aufgrund des mittleren (10.) Elements hätten sich die Teilbereiche 1-9 und 11-20 ergeben. In Wirklichkeit sind wir jedoch bei den Teilbereichen 1-13 und 15-20 angelangt.

Bevor Sie nun weiterlesen, sollten Sie den bisherigen Ablauf anhand der Zeilen 1-8 nochmals nachvollziehen. Wir wiederholen den Algorithmus in Kurzform:
1. Bereichsgrenzen (Unter- und Obergrenze) ermitteln; in Zeile 2 durch halbfett gedruckte Ziffern ersichtlich.
2. Mittleres Element willkürlich auswählen und dessen Wert zum Vergleichswert erklären; s. Zeile 2 rechts: Mitte: 10, V(ergleichwert): 72.
3. Wert des untersten Elements mit dem Wert des mittleren Elements vertauschen; s. Zeile 3: Ersttausch.
4. Ab dem zweituntersten Element nach rechts aufsteigend und ab dem obersten Element nach links absteigend jeweils ein Element suchen, das größer-gleich/kleiner-gleich als der Vergleichswert ist. Wenn die aufsteigende/absteigende Suche noch zu keiner Überlappung geführt hat, Werte der Elemente vertauschen und weitersuchen und -tauschen; s. Zeilen 4-7: normaler Tausch.
5. Andernfalls Wert des untersten Elements mit dem Wert des absteigend-überlappenden Elements vertauschen; s. Zeile 8: Letzttausch.

Nach der ersten Teilung

Das Quicksort-Motto lautet: Was man das erste Mal getan hat, kann man auch das zweite Mal tun („Rekursionsprinzip"). Folglich wenden wir exakt denselben Algorithmus nunmehr auf die zwei Teilbereiche 1-13 und 15-20 an. Dies hört sich jedoch leichter an, als es in der Programmierpraxis ist, denn wir müssen uns natürlich diejenigen Bereiche merken, die *momentan nicht* bearbeitet werden. In der Zeile 9 sind die Ziffern 1 und 13 halbfett markiert: Diese stellen die Begrenzungen des Bereichs dar, der zunächst *gemerkt* werden muß, und zwar auf dem Stack (= Stapel = Keller; im Applesoft-Demo-Programm als Pseudo-Stack-Array realisiert). In Zeile 9 rechts außen finden wir die Bemerkung
Stack, F:15, L:20,
die besagt, daß bei dem *momentanen* Bereich das 15. das unterste (F = first) und das 20. das oberste (L = last) Begrenzungselement sind.

Sie werden sich nunmehr drei Fragen stellen:
1. Wie tief wird geteilt? Antwort: Bis ein Teilbereich nur noch aus zwei oder einem einzigen Element besteht. Aus Gründen der Übersichtlichkeit wird bei unserer Tabelle die jeweils letzte Aufteilung, bei der das untere mit dem oberen Begrenzungselement identisch ist, nicht angezeigt. In diesen Fällen enthält der Stack vorübergehend 2 gleiche Element-Nummern.

2. Wann ist die Sortierung abgeschlossen? Antwort: Wenn alle Teilbereiche komplett abgearbeitet worden sind, d.h. wenn der Stack wieder leer ist.
3. Wie tief muß der Stack sein? Antwort: Wenn wir einen Pseudo-Stack-Array mit beispielsweise DIM SS% (30) anlegen und jeweils 2 Elemente für Unter- und Obergrenze eines „gemerkten" Teilbereichs benötigen, so können 15 „gemerkte" Teilbereiche auf den Stack geschoben werden. In der Praxis ist dies für ca. 1000 zu sortierende Elemente selbst bei „entarteten Folgen" stets ausreichend. Betrachten Sie hierzu die Zahl 1024: Einmal halbiert (= 2 Grenzwerte auf dem Stack), ergibt sich 512, danach 256, 128, 64, 32 usw. Da man sich bei nicht ganz „mittigen" Teilungen immer den größeren (= mehr Elemente umfassenden) Bereich merkt und mit dem kleineren (= weniger Elemente umfassenden) Bereich fortfährt, kann der Stack nie überlaufen. Jetzt wird uns auch der Grund für das mysteriöse „mittlere Element" klar: Theoretisch könnte man jedes *beliebige* Element des Teilungsbereichs als Vergleichswert herauspicken. Durch Selektion des mittleren Elements in Verbindung mit der Weiterbearbeitung der jeweils kleineren Bereiche befinden sich bei n (z.B. 1024) zu sortierenden Elementen im Stack niemals mehr als lb n (z.B. 10) gemerkte Teilbereiche (lb = Logarithmus zur Basis 2; 2 ↑ 10 = 1024).
Wenn Sie bei der Tabelle die Mitte-Zeilen 2, 10 und 28 betrachten, sehen Sie sehr gut die T-Kreuz-Formen der Bereichsteilungen, wobei der senkrechte T-Strich das jeweils endgültig positionierte Element anzeigt.

5. Anmerkungen zum Demo-Programm

Das gelistete Demo-Programm QUICK-.RANDOM erzeugt bei jedem Durchlauf eine neue Zufallszahlenfolge, während das zusätzlich auf die Peeker-Sammeldisk aufgenommene Demo QUICK.SPEZIAL drei „entartete" Zahlenfolgen vorführt. Ferner enthält die Sammeldisk die TASC-compilierbare Spezialversion QUICK-.TASC zur Geschwindigkeitsmessung.

Sentinel

Sentinel heißt Wachposten. So bezeichnet man im Englischen die Begrenzungswerte (Minimal- und Maximalwert). Beim aufsteigenden und absteigenden Durchsuchen eines Bereichs müssen Sie dafür Sorge tragen, daß die Suche nicht nach unten über den Anfangswert und nach oben über den Endwert „absaust".

Untere Begrenzung: Wenn wir das jeweils unterste Element durch den Vergleichswert ersetzen, kann die absteigende Suche niemals über das unterste Element hinausgehen, denn der Vergleichswert ist niemals kleiner als der Vergleichswert.
Obere Begrenzung: Wenn das oberste Element mindestens dem größten Wert des gesamten Arrays entspräche, könnte die aufsteigende Suche niemals über das oberste Element hinausgehen. Da es hierfür jedoch keine Garantie gibt, muß man entweder durch einen vorgeschalteten einmaligen Bubblesort-Durchgang das größte Element zum obersten Element machen oder – wie bei unserem Demo-Programm – ein weiteres Element (SN% + 1) mit dem maximalen Wert (hier 9999) belegen. Selbst wenn alle anderen Elemente den gleichen Wert 9999 hätten, würde die Sortierroutine niemals „absausen".

Unnötiger Erst/Letzttausch

Die Sentinel-Methode führt bei den kleinsten Teilbereichen zu einem unnötigen Erst- und Letzttausch. Betrachten wir hierzu in der Tabelle die Zeile 54 mit dem 2-Element-Bereich vom untersten (4.) Element (Wert 31) bis zum oberste (5.) Element (Wert 32). Zunächst wird das 4. Element als Vergleichswert erkoren. Da das 4. Element zugleich das unterste Element dieses Bereichs ist, wird das 4. Element mit dem 4. Element vertauscht (unnötiger Ersttausch). Da bei der nachfolgenden auf- und absteigenden Suche eine Überlappung stattfindet, wird erneut das 4. Element mit dem 4. Element vertauscht (unnötiger Letzttausch).
Man muß hierzu jedoch wissen, daß jede zusätzlich eingebaute Prüfung, die unnötige Vergleiche eliminiert, selbst in Form eines Vergleichs programmiert werden muß, der ebenfalls Zeit kostet, so daß eine Verfeinerung der hier vorgestellten Applesoft-Quicksort-Implementierung kaum zu Zeitgewinnen führen würde.

Sonderfälle

Das Demo funktioniert nur, wenn der zu sortierende Array mindestens 2 Elemente umfaßt. Ursprünglich für Zahlen-Arrays gedacht, läßt sich das Programm leicht für String-Quicksort umschreiben, indem man die Hilfsvariablen T und SV durch T$ und SV$ sowie den Array T (SN%) durch T$ (SN%) ersetzt.

Geschwindigkeit

Abschließend sei darauf hingewiesen, daß die TASC-compilierte Version von QUICK.TASC zum Sortieren eines Integer-Arrays mit 1000 Elementen ca. 17 Sekunden benötigt. Dagegen bringt es die nicht-compilierte Applesoft-Version QUICK.TASC nur auf 215 Sekunden. Der TASC-Compiler bewirkt hier also eine 12fache Geschwindigkeitssteigerung. Die Assembler-Quicksort-Routine von Harald Grumser bewirkt nochmals eine 5fache Steigerung gegenüber der bestmöglichen TASC-Version.

Kurzhinweise

1. Zweck:
Demonstration von Quicksort.
2. Konfiguration:
Apple IIe/c/+ mit einer 80-Zeichenkarte, die die Applesoft-Befehle INVERSE/NORMAL erkennt; DOS 3.3 oder ProDOS.
3. Test:
RUN QUICK.RANDOM
RUN QUICK.SPEZIAL
RUN QUICK.TASC
RUN QUICK.DISK
4. Sammeldisk:
QUICK.RANDOM
QUICK.SPEZIAL
QUICK.TASC
QUICK.DISK

DB-MEISTER

Adreß- und Schemabriefprogramm

Der DB-Meister ist ein in Assembler geschriebenes, ungewöhnlich schnelles, unkompliziertes und zugleich „narrensicheres" Adreß-, Datei- und Schemabriefprogramm.

Technische Daten
- Recordlänge bis zu 230 Zeichen
- 560 bis 1000 Records pro Datendiskette
- Maximal 25 Felder pro Record
- Suche nach 3 Indexfeldern
- Ausdruck der Dateien als Etiketten, Listen und Schemabriefe (mit Felder- und Tastatureinschüben an beliebigen Stellen des Formbriefes)
- normal kopierbare Programmdiskette, unterteilt in Hauptprogramme und diverse Hilfsprogramme
- einsatzfähig auf Apple IIe und IIc mit 2 Drives (1 Drive ebenfalls möglich)

Gesamtpreis 290,– (2 Disketten + gedrucktes Manual)

U. Stiehl
c/o Dr. A. Hüthig Verlag
Postfach 10 28 69 · 6900 Heidelberg

QUICK.RANDOM

Erzeugt tabellarisches Sortierprotokoll von 20 Zufallszahlen, die mit jedem RUN neu erzeugt werden.

1. Laden mit LOAD QUICK.RANDOM
2.1. Entweder 80-Zeichenkarte mit PR#3 einschalten oder
2.2. in Zeilen 760 und 780 Druckbefehle für Fettdruck/Normaldruck statt INVERSE/NORMAL einsetzen und Drucker mit PR#1 einschalten.
3. Jeweils mit RUN neue Tabelle generieren.

```
100 HOME : GOSUB 760:
    PRINT "Quicksort-Demo von U.Stiehl";:
    GOSUB 780: PRINT
110 REM 20 Zufallszahlen
120 SN% = 20: DIM T(SN% + 1)
130 Z = 0: REM Zeilenzähler
140 GOSUB 800: FOR X = 1 TO SN%:
    T(X) = INT (90 * RND (1)) + 10:
    PRINT T(X);" ";: NEXT X:
    PRINT "unsortiert"
150 GOSUB 190: REM Sortieren
160 GOSUB 800: FOR X = 1 TO SN%:
    PRINT T(X);" ";: NEXT X:
    PRINT "sortiert"
170 END

    Initialisierung

180 REM T(SN%+1)
    Inkrement-'Sentinel' = maximaler Wert
190 T(SN% + 1) = 9999
200 REM Array von SF%=First Element
    bis SL%=Last Element sortieren
210 SF% = 1:SL% = SN%
220 DIM SS%(30):SC% = 0

    Bereich ermitteln ("Partition")

230 REM Wenn SF%>=SL% dann Stack prüfen
240 IF SF% > = SL% THEN 520
250 REM Dekrement-Zeiger SD%
    vorläufig auf SL%+1 setzen
260 SD% = SL% + 1
270 REM Inkrement-Zeiger SI%
    auf SF% setzen
280 SI% = SF%
290 REM Mittlere Positon SM% zwischen
    SF% und SL% ermitteln
300 SM% = INT ((SL% - SF%) / 2) + SF%:
    GOSUB 560: REM Anzeige

    Ersttausch

310 REM SM%-Wert zum Vergleichwert SV
    machen und SM%-Wert mit SF%-Wert
    tauschen; Ersttausch;
    Dekrement-'Sentinel' = minimaler Wert
320 SV = T(SM%):T(SM%) = T(SF%):
    T(SF%) = SV: GOSUB 620: GOSUB 700:
    REM Anzeigen

    Inkrement und Dekrement

330 REM SI% erhöhen, solange T(SI%) < ST
340 SI% = SI% + 1: IF T(SI%) < SV
    THEN 340
350 REM SD% vermindern, solange T(SD%) > SV
360 SD% = SD% - 1: IF T(SD%) > SV
    THEN 360
370 REM Überkreuzen sich SI% und SD%?
380 IF SD% < = SI% THEN GOTO 430

    Normaler Tausch

390 REM Wenn nein, momentanen SI%-Wert
    mit SD%-Wert tauschen;
    normaler Tausch
400 TT = T(SI%):T(SI%) = T(SD%):
    T(SD%) = TT: GOSUB 740: REM Anzeigen
410 GOTO 340

    Letzttausch

420 REM Wenn ja, SF%-Wert durch SD%-Wert
    und SD%-Wert durch SV ersetzen;
    Letzttausch
430 T(SF%) = T(SD%):T(SD%) = SV:
    GOSUB 720: REM Anzeigen

    Stack-Routine

440 REM Größeren Teilabschnitt ermitteln
450 IF SD% - SF% < SL% - SD% THEN 480
460 REM SF% und SL% des neuen
    Bereichs auf Stack schieben
470 SS%(SC% + 1) = SF%:
    SS%(SC% + 2) = SD% - 1:
    SF% = SD% + 1: GOTO 500
480 SS%(SC% + 1) = SD% + 1:
    SS%(SC% + 2) = SL%:SL% = SD% - 1
490 REM Stack-Counter SC% um 2 erhöhen
500 SC% = SC% + 2: GOTO 240
510 REM SF% und SL% des letzten
    Bereichs vom Stack holen
520 IF SC% > 0 THEN SL% = SS%(SC%):
    SF% = SS%(SC% - 1):SC% = SC% - 2:
    GOTO 240
530 REM Sortieren beendet,
    wenn SC% = 0 ist
540 RETURN

    Tabelle anzeigen

550 REM *** Anzeigen ***
560 IF SC% = 0 GOTO 610
570 GOSUB 800: FOR Y = 1 TO SN%
580 FOR X = 1 TO SC%:
    IF SS%(X) = Y THEN GOSUB 760:
    X = SC%
590 NEXT : IF Y < 10 THEN PRINT "0";
600 PRINT Y;: GOSUB 780: PRINT " ";:NEXT:
    PRINT "Stack,F:";SF%;",L:";SL%
610 RETURN
620 GOSUB 800: FOR Y = 1 TO SN%
630 IF Y = SF% THEN GOSUB 760
640 IF Y < 10 THEN PRINT "0";
650 PRINT Y;
660 IF Y = SL% THEN GOSUB 780
670 PRINT " ";
680 NEXT : PRINT "Mitte ";SM%;",V:";SV
690 RETURN
700 GOSUB 800: FOR Y = 1 TO SN%:
    IF Y = SM% OR Y = SF% THEN GOSUB 760
710 PRINT T(Y);: GOSUB 780: PRINT " ";:
    NEXT Y: PRINT "Ersttausch": RETURN
720 GOSUB 800: FOR Y = 1 TO SN%:
    IF Y = SF% OR Y = SD% THEN GOSUB 760
730 PRINT T(Y);: GOSUB 780: PRINT " ";:
    NEXT Y: PRINT "Letzttausch": RETURN
740 GOSUB 800: FOR Y = 1 TO SN%:
    IF Y = SI% OR Y = SD% THEN GOSUB 760
750 PRINT T(Y);: GOSUB 780: PRINT " ";:
    NEXT Y: PRINT "Tausch": RETURN
760 INVERSE : GOTO 770: REM Dummy!!!
    PRINT CHR$ (27); CHR$ (71);
770 F = 1: RETURN
780 IF F = 1 THEN F = 0: NORMAL :
    GOTO 790: REM Dummy!!!
    PRINT CHR$ (27); CHR$ (72);
790 RETURN
800 Z = Z + 1: IF Z < 10 THEN PRINT "0";
810 PRINT Z; ": ";: RETURN

    Variablen

820 REM SN%=Dimension des Zahlenarrays
830 REM SF%=First=Anfang des Bereichs
840 REM SM%=Mitte des Bereichs
850 REM SL%=Last=Ende des Bereichs
860 REM SI%=Inkrement-Zeiger
870 REM SD%=Dekrement-Zeiger
880 REM SS%=Stack=Stapel
890 REM SC%=Stack-Counter
900 REM SV=Vergleichshilfsvariable
910 REM TT=Tauschhilfsvariable
920 REM T(SN%)=Zahlenarray
```

QUICK.SPEZIAL

Das analoge Programm QUICK.SPEZIAL, das sich auf der Peeker-Sammeldisk befindet, demonstriert 3 Spezialfälle von zu sortierenden Arrays:
a) aufsteigend vorsortierte Elemente
b) absteigend vorsortierte Elemente
c) völlig gleiche Elemente.

Apple und IBM kompatible Computer

16K, Z80, Diskcontroler je 75,–
80 Zeichenkarte mit Softswitch
2 Zeichensätze 149,–
Motherboard 48K ohne Firmware . . 419,–
Erphi-controler mit Autopatch . . . 300,–
Siemenslaufwerk F 122 515,–
TEAC FD-55B 2 × 40 Track 375,–
TEAC FD-55F 2 × 80 Track 395,–
FD4 Spezialcontroler für Laufwerke
mit bis zu 2 × 80 Track 120,–
Drucker Star SG 10 940,–
Monochrome Monitore . . . . . ab 375,–
Farbmonitore . . . . . ab 998,–
Tastaturen für IBM und Apple . ab 330,–
Versand nur per Nachnahme oder Vorkasse.
Weiteres Zubehör für Apple und IBM gegen frankierten Rückumschlag.
Preissenkung:
128K Karte (Saturn kompatibel) . **248,–**
Preissenkung 4164-200 ns
Mindestabnahme 20 Stck. 2,50
Zusatzkarten und Motherboard ausnahmslos deutsche Fertigung mit ausgesuchten Bauteilen.

Ulf Mohwinkel Electronic
Berliner Straße 73 Pf: 250 166
5090 Leverkusen Fettehenne
Telefon 02 14/9 37 81 od. 9 50 60

Preisliste kostenlos
Katalog DM 2,–
in Briefm.

Unser Dezember-Angebot:

Profimax IIe, APPLE IIe kompatibel, IBM-Look **nur 1398.–**
Profimax III, APPLE II+ kompatibel, 2 Prozessoren IBM-Look **nur 1098,–**
Chinon-Laufwerk 051A **nur 398.–**
Video-Digitalisierer incl. Software **nur 298.–**
STAR sg-10, original, mit Serien-Nr. **nur 998.–**
Centronics-Interface, grafikfähig **nur 118.–**

D.O.S Computersysteme
der Partner für Schule, Hochschule, Forschung, Fertigung

Am Kühnbach 42, 7170 Schwäb. Hall 11
Tel. 07 91/5 17 36

Berlin im Apple II+ 384 x 288 Bildpunkte.

VIDEO-1000

DIGITIZER ZUM APPLE IIC

INTERFACE+ SOFTWARE **295,-DM**

* Auflösung 384x288 Bildpunkte
* für TV, Recorder und Kamera
* Aufnahmezeit 20 msec
* Umrechnung auf HIRES-Seite

Erweiterte Software z.B. Bilder mit bis zu 300.000 Bildpunkten, 7 Grausturen, Doubble Hires, Kurzfilmerstellung etc auf Anfrage.

256 K RAM-Karte mit Software .....495,- DM
192K RAM-Karte mit Software .....395,- DM
16 K AKKU-RAM-Karte mit Software .....175,- DM
120K AKKU-RAM-Karte mit Software .....595,- DM

Demodisk gegen Einsendung von 10 DM oder V-Scheck. Info gratis. Versand p.NN. oder beim Fachhändler.

Ing.Büro M.Fricke, Neue Str.13a, 1000 Berlin 37
Telefon: 030 / 801 50 52

QUICK.DISK
Externes Sortieren

Wenn eine Datei so groß ist, daß nicht einmal die Sortierfelder (= Schlüssel) geschweige denn die Datensätze selbst in den Speicher eingelesen werden können, muß man extern auf der Diskette, Festplatte oder RAM-Disk sortieren. Die nachfolgende QUICK.DISK-Routine gestattet das externe Sortieren einer beliebig großen Random-Datei (2 ↑ 15 Sätze). Das Demo selbst, das unter DOS 3.3 und ProDOS funktioniert, ist jedoch auf maximal 9999 4stellige Zahlen begrenzt, was für die 64K-Karte-RAM-Disk ausreicht = 50.000 Bytes. Von der Verwendung eines normalen Diskettenlaufwerks muß dringend abgeraten werden, denn dies würde bei 10.000 Zahlen Tage dauern. Die Sortierroutine eignet sich demnach nur entweder für eine RAM-Disk oder eine Festplatte.

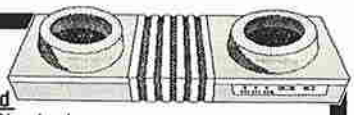
```
10 TEXT : HOME : INVERSE : PRINT "QUICKSORT-DISK": NORMAL
12 SB = 1: PRINT : INPUT "Wieviele Zahlen: ";SN:
   SN = INT (SN): IF SN < 2 OR SN > 9999 GOTO 12
14 PRINT : PRINT SN;" Zufallszahlen erzeugen..."
16 PRINT CHR$ (4)"OPEN Z,L5": PRINT CHR$ (4)"WRITE Z,R0":
   PRINT SN
18 FOR X = SB TO SN:TT = INT (9000 * RND (1)) + 1000:
   P1 = P1 + TT
20 PRINT CHR$ (4)"WRITE Z,R"X: PRINT TT: NEXT X
22 PRINT CHR$ (4)"WRITE Z,R"X: PRINT 9999:
   PRINT CHR$ (4)"CLOSE Z"
24 PRINT : PRINT "Stoppuhr ";: GET X$: PRINT CHR$ (7):
   GOSUB 36:X = PEEK (49168)
26 IF PEEK ( - 16384) < 127 THEN CALL - 1059: GOTO 26
28 PRINT : PRINT "Prüfsumme ermitteln...":
   PRINT CHR$ (4)"OPEN Z,L5": PRINT CHR$ (4)"READ Z,R0":
   INPUT SN
30 TV = 0: FOR X = SB TO SN: PRINT CHR$ (4)"READ Z,R"X:
   INPUT TT:P2 = P2 + TT: IF TV > TT THEN
   PRINT CHR$ (4)"CLOSE Z": PRINT "Fehler": END
32 TV = TT: NEXT : PRINT CHR$ (4)"CLOSE Z":
   PRINT P1: PRINT P2
34 END

   Unterroutine für externes Quicksort

36 PRINT "Sort-Anfang":
   SF = SB:SL = SN: DIM SS(30):SC = 0
38 PRINT CHR$ (4)"OPEN Z,L5"
40 ON SF < SL GOTO 44: IF SC > 0 THEN
   SL = SS(SC):SF = SS(SC - 1):SC = SC - 2: GOTO 40
42 PRINT CHR$ (4)"CLOSE Z": PRINT "Sort-Ende": RETURN
44 SD = SL + 1:SI = SF:SM = INT ((SL - SF) / 2) + SF
46 PRINT CHR$ (4)"READ Z,R"SM: INPUT TV:
   PRINT CHR$ (4)"READ Z,R"SF: INPUT TT:
   PRINT CHR$ (4)"WRITE Z,R"SM: PRINT TT:
   PRINT CHR$ (4)"WRITE Z,R"SF: PRINT TV
48 SI = SI + 1: PRINT CHR$ (4)"READ Z,R"SI:
   INPUT TI: IF TI < TV THEN 48
50 SD = SD - 1: PRINT CHR$ (4)"READ Z,R"SD:
   INPUT TD: IF TD > TV THEN 50
52 IF SD > SI THEN PRINT CHR$ (4)"WRITE Z,R"SI:
   PRINT TD: PRINT CHR$ (4)"WRITE Z,R"SD:
   PRINT TI: GOTO 48
54 PRINT CHR$ (4)"WRITE Z,R"SF: PRINT TD:
   PRINT CHR$ (4)"WRITE Z,R"SD: PRINT TV
56 IF SD - SF < SL - SD THEN
   SS(SC + 1) = SD + 1:SS(SC + 2) = SL:
   SL = SD - 1:SC = SC + 2: GOTO 40
58 SS(SC + 1) = SF:SS(SC + 2) = SD - 1:
   SF = SD + 1:SC = SC + 2: GOTO 40
60 REM Für 9999 Zahlen ca. 1 Stunde Sortierzeit
   unter ProDOS + RAM-Disk + Accelerator
```

Bemerkung:

Die Sortierzeiten wären in der Praxis unvertretbar. Deshalb kommen anstelle des externen Sortierens in der Regel Misch- oder Merge-Programme zum Einsatz. Hierüber wird in einem gesonderten Peeker-Aufsatz berichtet.



AKUSTIK-KOPPLER - Dataphon s21d
300 Baud Modem, nach CCITT V.21 Standard, m. FTZ-Nr. 18.13.1917.00, Gebühren- und anmeldefrei, V24/RS-232 Standard-Schnittst. nur DM 298,00

TELEKOMMUNIKATIONS - KOMPLETT - PAKET
geeignet für Apple //+ und Apple //e:
1 Dataphon s21d,
1 Anschlußkabel: V.24 zum Apple II-Game-I/O,
1 Terminalprogramm: "HIB Modem-Transfer" nur DM 349,00

Chinon-Laufwerk (Testbericht in Peeker 5/85)
für Apple //+ und Apple //e anschlußf. im Gehäuse DM 498,00
w.o. jedoch für Apple //c DM 569,00

TOSHIBA Spitzenlaufwerke zum Superpreis!
ND 06-D, 2 x 80 Track, 640 K-Byte formatiert DM 498,00

DISK-DOPPEL-STATION (APPLE //+, APPLE //e)
2 x ND 06-D im Geh. + Auto-Patchcontr., 1,2 MB DM 1598,00

AUTO-PATCH-CONTROLLER DM 298,00

IC-Test-Karte (Testet ca. 500 verschiedene IC's) DM 398,00

BROTHER-Matrixdrucker, die Super-Drucker!
M-1009 (Matrixdrucker, RS-232 + Centronics) DM 429,00
M-1009 anschlußfertig an:
Apple //c (mit Drucker-Kabel) DM 529,00
Apple //e (mit Graphik-Interface und Kabel) DM 629,00

Alle Preise inclusive der gesetzlichen Mehrwertsteuer.
Berechnung der Versandkosten erfolgt nach Entfernung und Gewicht.
Fordern Sie noch heute unsere Gratispreisliste an!Wiederverkäufer bitte nur schriftlich anfragen (Kopie der Gewerbeanmeldung beilegen!).

hib
HIB Computerladen
Äuß. Bayreuther Str. 72 - Telefon: 0911 / 515 939
Postfach 21 01 25 - Telex: 17 - 911 8253
8500 Nürnberg 21 - Teletex: 2526 - 911 82 53

Die ProDOS-Analyse

Version 1.0.1, 1.0.2, 1.1.1

von Arne Schäpers

1985, 470 S., kart.,
DM 68,–
ISBN 3-7785-1134-3

„Die ProDOS Analyse" ist die umfangreichste und detaillierteste Darstellung, die jemals ein Apple-Betriebssystem erfahren hat. Wer die „Innereien" von ProDOS bis zum letzten Byte, z. T. bis ins letzte Bit kennenlernen möchte, braucht dieses Buch. Das komplette Betriebssystem (Urlader, MLI, Disk-Driver, RAM-Disk-Driver und Uhr-Routine) mit Ausnahme des BASIC-SYSTEM wird mit umfangreichen Kommentaren und Übersichtstabellen disassembliert. Dabei werden alle bisherigen Versionen von 1.0.1 bis 1.1.1 berücksichtigt. „Die ProDOS-Analyse" beschreibt erstmals auch mehrere Programmierfehler, die bis in die neueste Version zu finden sind. Auch die nicht im „Technical Reference Manual" aufgeführten Eigenschaften von ProDOS werden analysiert und beschrieben, z. B. die vertrackten eingebauten Testroutinen zur Identifikation der verschiedenen Apple-II-Modelle und eventueller Nachbaugeräte. Programmierer, die ProDOS versionsabhängig „patchen" möchten, erhalten hier den genauen Überblick, wo was geändert werden muß, damit dies keine negativen Konsequenzen hat. Durch die minutiöse theoretische Sezierung von ProDOS eröffnen sich völlig neue programmierpraktische Perspektiven.

Dr. Alfred Hüthig Verlag
6900 Heidelberg · Postfach 10 28 60

Ausgabe und Eingabe mit

TYPETERM®

im Slot Ihres

APPLE II/IIe

Das bedeutet: Computer-textverarbeitung von der Schreibmaschinentastatur! Steckerfertig ohne Umbau.

TYPETERM-Interface **DM 479,–**
für alle BROTHER-Typenrad-schreibmaschinen ab CE-51

Paketpreis: **DM 1348,–**
Schreibmaschine CE-51 mit TYPETERM

CE-61 mit TYPETERM ..... DM 1737,–
CE-70 mit TYPETERM ..... DM 2758,–
EM-80 mit TYPETERM ..... DM 2037,–
TYPETERM-Kit für CE-50 ..... DM 468,–

TYPETERM – ein starkes Interface für starke Maschinen! Alle Cursor- und Ctl-Befehle. 4k ROM auf der Karte für DOS, PRODOS, CP/M, PASCAL. 2 Zeichensätze verfügbar z. B. deutsch u. ASCII. Alle Features: Hoch-/Tiefstellen, autom. Unterstreichen, var. Zeichen und Zeilenabst., autom. Papierzuführung usw. Ausführl. Handbuch vorab: 10,– DM auf Konto 14770-306 PGiroA Han (Anrechnung).

TYPETERM – ein Produkt von

interkom electronic
Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 05139-87393

Ausgabe mit

TYPETERM® JUNIOR

im Slot Ihres

APPLE II/IIe

Paketpreis **DM 899,–**
Schreibmaschine AX-10 mit Interface TYPETERM JUNIOR, steckfertig.

brother
QUALITÄT AUS ERSTER HAND.

TYPETERM JUNIOR mit AX-10 – unser besonders günstiges Gespann, ebenfalls steckfertig. Mit TYPETERM JUNIOR kann die AX-10 mehr. Sie wird zum vollwertigen Typenraddrucker für Ihren Apple:

- 3 verschiedene Schriftstärken
- Automatisches Unterstreichen
- 2 Zeichensätze z.B. deutsch u. ASCII
- 2 Zeichenabstände
- 2k ROM auf der Karte für Ausgabe unter DOS, PRODOS, CP/M u. PASCAL.

TYPETERM JUNIOR – ein Produkt von

interkom electronic
Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 05139-87393

ACS UNIVERSAL KEYBOARDS

Modell AN95FE... DM 448,– ohne MwSt. (DM 510,72 incl. MwSt.)
Die KEYBOARDS SPEZIELL angepaßt für den APPLE IIe
Händleranfragen erwünscht

- FLEXIBEL — Jede Taste frei im EPROM programmierbar in bis zu 8 Ebenen im mitgelieferten EPROM
- PROFESSIONELL — Für Anwender mit gehobenen Ansprüchen
- ERGONOMISCH — Nach DIN ULTRAFLACH gestaltetes stabiles Gehäuse
- KOMPLETT — Tastatur, Gehäuse und Kabel fertig montiert und getestet. Durch Spezial-Kabel und Spezial-EPROM sofort einsteckfertig.
- KOMPAKT + FLACH — Durch Einsatz von „SIEMENS" Flachtasten-modulen

ACS
gesellschaft für computersteuerungen und datentechnik mbh
D-4930 Detmold ★ Alter Mühlenweg 5
Telefon 0 52 31/41 76 ★ Telex 9 35 660 acs d

Achtung! Zusatzprogramm für *AppleWorks*

DIESE ANZEIGE

wurde vollständig mit *AppleWorks* gedruckt. Dazu benötigen Sie nur noch den *DMP Charger*, der außerdem noch 20 weitere Zeichensätze für Sie bereit hält.

CAPITALS, *Italic*, Schmasch, Schreib,
АВДЕФГНЪКЛМП, ОПЕРСВХУZ

ATHENS LONDON ZEIT

Auch vom *WordStar* und anderen Programmen können diese Zeichen benutzt werden. Sie können die Zeichen verändern, oder auch neue gestalten. Das Programm arbeitet mit dem ✿ //e (128k) oder ✿ //c und mit dem ✿ Imagewriter, auch FX 80, Panasonic, Okidata.
Preis: 198,- incl. MwSt. und ausführlichem Handbuch. *Infoblatt kostenlos.* Versand gegen offene Rechnung möglich

Norbert Hunstig
Nottulner Landweg 81
D-4400 Münster
Tel.: 02534/ 7036 Telex: 892 496

auch bei Pandasoft und Intus erhältlich.

Semjan presents...

Cirtech Produkte für Apple II

• CP/M Plus System für Apple //c

CP/M Plus System mit Betriebssystem, Z80H (8MHz), 128K RAM, Maus-Funktion, Drucker-Spooler mit 12K RAM, Tastatur-Spooler, Bildschirm Dump zu jederzeit, Kompatibel zu CP/M 2.2 u. 2.23. Einsatz aller CP/M Sprachen und Programme (MBASIC, WORDSTAR etc.).

K10 Apple //c CP/M Plus System Modul DM 998,00
K11 Apple //c CP/M Plus System Modul und WORDSTAR/ MAILMERGE DM 1498,00

• Champion Drucker Karte

Parallele Text- u. Graphik-Druckerkarte, 16 o. 64K RAM Puffer, 40/80 Z. Dump, Einsatz von DOS, PRODOS, PASCAL, CP/M und auch Appleworks, volle Apple //e Graphik, Serieller Ausbau möglich.

K32 Apple //+,e Champion Interface 16K RAM DM 459,00
K33 Apple //+,e Champion Interface 64K RAM DM 599,00

• 80 Zeichen Karte mit 64K RAM

80-Zeichen Karte mit 64K RAM, voll Apple //e Graphik fähig.

K51 Apple //e 80-Zeichen Karte mit 64K RAM DM 359,00

• 1Megabyte RAM Karte

100 % Kompatibel mit PRODOS (Appleworks), DOS, PASCAL 1.3 und CIRTECH CP/M Plus System, max. 6 Megabyte pro Apple //+,e, Einsatz als RAM-Disk in jedem Slot möglich.

K70 Apple //+,e Flipper Karte mit 1MB RAM. DM 1798,00

Händleranfragen willkommen!

M. Semjan Computer Systeme
Postfach 90 01 64 · 6000 Frankfurt/M 90
Tel. 069-70 18 53 · Mo-Fr 10.30-15 Uhr

Quicksort

Eine superschnelle Applesoft-Erweiterung

von Harald Grumser

In vielen Anwendungen taucht das Problem auf, Daten beliebiger Art zu sortieren. Bei geringerem Umfang der Datenmenge bietet sich ein Applesoft-Programm an, wie es in dem Artikel „Wie funktioniert Quicksort?" in diesem Heft beschrieben wird (die Lektüre dieses Beitrags wird hier vorausgesetzt). Diese Vorgehensweise erfordert jedoch die Anpassung des Programms an den jeweiligen Datentyp und benötigt bei einer großen Zahl von Array-Elementen trotz des intelligenten Sortieralgorithmus eine beträchtliche Zeit. Abhilfe schafft hier der Übergang zu einem Assemblerprogramm, das darüber hinaus alle drei Datentypen von Applesoft (Real-Zahlen, Integer-Zahlen und Strings) unterstützt. So ist es möglich, 1000 Real-Zahlen in einer Zeit von unter 3 Sekunden zu sortieren. Außerdem wird bei Strings eine Sortierung nach Duden vorgenommen.
Das Programm **QUICKSORT** wird als Ampersand-Utility in das eigene Applesoft-Programm eingebunden, nachdem es durch
BRUN QUICKSORT
(unter DOS 3.3, nicht unter ProDOS!) installiert wurde. Dieser Aufruf sollte stets am Programmanfang erfolgen, da HIMEM verändert wird. Der Sortiervorgang wird dann eingeleitet durch den Befehl „&Array", wobei Array das zu sortierende Feld bezeichnet, also z.B. XWERT, PUNKTE% oder NAME$ (siehe **QUICKSORT-.DEMO**).

1. Die Grundlagen

Bei der Programmierung eines umfangreicheren Programms erweist es sich als vorteilhaft, zunächst einige Teilaspekte zu untersuchen, um dann das ganze Problem in Angriff zu nehmen (sog. Bottom-Up-Programmierung). Dieser Weg soll auch bei der Erläuterung dieses Programms beschritten werden.

Aufbau von Array-Variablen

Array-Variablen (auch indizierte oder Feld-Variablen genannt) werden wie alle Variablen hinter dem Applesoft-Programm abgelegt. Während einfache Variablen stets 7 Bytes einnehmen, hängt der Umfang eines Feldes von drei Faktoren ab. Zunächst kann ein Array bis zu 255 Dimensionen umfassen, wobei jede Dimension wiederum aus bis zu (theoretisch) 65535 Elementen bestehen kann. Die Größe eines Elements hängt dann von dem Datentyp ab:

– Integer-Zahlen belegen je zwei Bytes im Speicher, die die Zahl (im Bereich -32767 bis 32767) repräsentieren.
– String-Variablen belegen je drei Bytes. Das erste Byte gibt die String-Länge wieder, während die beiden nächsten Bytes auf die eigentliche Zeichenkette, die sich im String-Pool unterhalb von HIMEM oder im Applesoft-Programm selbst befindet, verweisen.
– Real-Zahlen werden durch fünf Bytes gespeichert, ein Byte Exponent und vier Bytes Mantisse.

Jedes Feld wird durch einen Vorspann (Header) gekennzeichnet. Die ersten beiden Bytes geben Auskunft über den Namen, wobei die beiden Bit 7 den Variablentyp bestimmen. Danach folgt ein Zwei-Byte-Offset, der auf das nächste Feld verweist. Das nächste Byte gibt die Anzahl der Dimensionen wieder, woran sich für jede Dimension je zwei Bytes anschließen, die die Anzahl der Elemente pro Dimension beinhalten. Erst dann folgen die einzelnen Elemente des Arrays (weiter Informationen hierzu sind der Applesoft BASIC Programmieranleitung, Anhang I zu entnehmen).
Da das Programm nur eindimensionale Felder unterstützt, entfällt die umständliche Berechnung der Speicherabbildungsfunktion (diese Funktion weist jeder Element-Koordinate des n-dimensionalen Feldes eine lineare Speicheradresse zu).

Die Auswertung des Arrays

Beim Einsprung in die Quicksort-Routine steht der Programmzeiger (Textpointer) auf dem ersten Zeichen des Array-Namens. Der Applesoft-Interpreter enthält eine Routine, die diesen Namen auswertet und einen Zeiger auf den Beginn des angegebenen Feldes richtet. Diese Routine namens **PTRGET** kann durch das sog. SUBFLAG gesteuert werden, um z.B. zu verhindern, daß das Feld neu angelegt wird. Nach der Bestimmung der Speicheradresse des Arrays muß zunächst der Datentyp bzw. die Länge eines Elements mit Hilfe der Speicherstellen VALTYP und INTFLAG ermittelt werden, da bei der späteren Bearbeitung stets die Länge berücksichtigt werden muß.
Danach erfolgt eine Überprüfung der Dimension, um gegebenenfalls via **BSSERR** die Meldung „BAD SUBSCRIPT ERROR" auszugeben, falls es sich um ein mehrdimensionales Feld handelt.
Mit der sich daran anschließenden Berechnung der Feldlänge wird der Anfangs- und Endwert (START und END) gesetzt und somit das Intervall für den ersten Rekursionsschritt festgelegt.

Vergleich zweier Elemente

Ein wesentlicher Bestandteil des Sortierens besteht im Vergleich zweier Elemente. Hierzu wird ein Element in den Fließkomma-Akkumulator FAC übertragen, im Fall von FP-Variablen durch die ROM-Routine **MOVFM** und ansonsten durch einen eigenen Programmteil. Es bietet sich an, das mittlere Element des Intervalls, das während eines ganzen Rekursionsschrittes als Grundlage aller Vergleiche dient, dort abzulegen.

Der Vergleich muß abhängig vom Datentyp erfolgen. Real-Zahlen können durch **FCOMP** (Einsprung bei FCOMP1) verglichen werden, während bei Integer-Zahlen eine eigene Routine die Verarbeitung beschleunigt, da der Interpreter eine Typumwandlung in Real-Zahlen vornehmen würde, was unnötige Zeit kostet. Der Vergleich wird durch das Vorzeichen-Bit (Bit 15) erschwert, da bei negativen Zahlen die Logik umgekehrt werden muß und bei Zahlen mit unterschiedlichen Vorzeichen nur das Vorzeichen als Kriterium herangezogen werden darf.
Bei String-Variablen kann ebenfalls nicht auf eine Interpreter-Routine zurückgegriffen werden, da der Vergleich der einzelnen ASCII-Werte keine Sortierung nach Duden ergeben würde.

Duden-Normierung

Die Duden-Normierung sieht folgende Abweichungen gegenüber dem Applesoft-Vergleich von Strings vor:
– Groß- und Kleinbuchstaben sind äquivalent.
– Zahlen und Sonderzeichen haben keinen Sortierwert.
– Umlaute (ä, ö, ü) werden wie die entsprechenden Selbstlaute (a, o, u) behandelt.
Die Sonderstellung des Eszett wird hier nicht berücksichtigt, so daß das „ß" nach „z" eingeordnet wird.
Die Routine, die diese Anpassung vornimmt, wird für beide zu vergleichenden Strings benutzt. Ein kleiner Trick erspart hier einige Zyklen an Rechenzeit. Statt eines Aufrufs als Unterprogramm wird der Programmablauf durch das V-Flag gesteuert.

Die Verwaltung der Elemente

Grundsätzlich bestehen zwei Möglichkeiten, den Zugriff auf die einzelnen Elemente zu gestalten:
– Es wird der jeweilige Index verwaltet und vor einem Zugriff auf ein Feldelement dieser Index mit der Länge eines Elements multipliziert und die Anfangsadresse des Arrays addiert, um zu der entsprechenden Speicherposition zu gelangen. Diese Verwaltung erweist sich als sehr einfach bei der Bestimmung des mittleren Elements, kostet jedoch viel Zeit.
– Die einzelnen Elemente werden direkt über ihre Speicherposition verwaltet, wodurch sich der Zugriff sehr vereinfacht. Diese Möglichkeit bereitet jedoch Schwierigkeiten bei der Bestimmung des mittleren Elements, da eine Halbierung des Intervalls bei ungeradzahliger Anzahl von Elementen einen Zeiger auf die Mitte eines Eintrags erzeugen würde. Daher muß in diesem Fall ein Element von der Intervallänge subtrahiert werden, was bei der Bestimmung MIDDLE = (START + END) / 2 einer Abrundung entspricht. Die Festlegung, ob es sich um eine gerade oder ungerade Anzahl von Elementen handelt, ist nicht ganz einfach, da sie nur über den START- und END-Wert in Abhängigkeit von der Elementlänge (2, 3 oder 5 Bytes) erfolgen kann. Die Lösung sieht dementsprechend kompliziert aus: Sind START- und END-Wert beide gerade oder ungerade (zu erkennen am gleichen Bit 0), muß bei ungerader Elementlänge (3 oder 5 Bytes) eine Anpassung erfolgen. Sind START- und END-Wert beide in der zweiten Stelle gerade oder ungerade (zu erkennen am gleichen Bit 1), muß bei gerader Elementlänge (2 Bytes) eine Anpassung erfolgen.
Der zweiten Möglichkeit wurde wegen ihrer kürzeren Ausführungszeit der Vorzug gegeben.

2. Die Programmausführung

Das Verständnis des Quicksort-Algorithmus wird für die weiteren Ausführungen vorausgesetzt.
Die Vorgehensweise des Assemblerprogramms unterscheidet sich nur in wenigen Punken von dem Applesoft-Programm des o.g. Artikels.
Erst- und Letzttausch entfallen wie auch die dazu analoge Einrichtung eines Sentinels am Ende des Feldes, da dies zu einer Überschreibung des eventuell nächsten Arrays führen würde.
Die Bildung der Intervalle gestaltet sich ebenfalls etwas anders. Am Ende eines Rekursionsschrittes werden linker und rechter Zeiger vertauscht, um sicherzustellen, daß sich die Intervallgrenzen nicht überschneiden. Der eventuelle Austausch zweier gleicher Zeiger wird dabei in Kauf genommen.
Somit kann jedoch nicht mehr durch die Überschneidung der Zeiger das Ende eines Rekursionszweiges erkannt werden. Als Kriterium zum Abbruch des Intervalls wird daher die Intervallänge herangezogen. Falls weniger als drei Elemente vorliegen (d.h. erstes und mittleres Element sind wegen der Abrundung beim Halbieren gleich), wird dieser Ast beendet und ein neues Intervall vom Stack geholt.
Die gravierendste Unterscheidung liegt in der Bestimmung des auf den Stack zu schiebenden Intervalls. Wird stets das kleinere Intervall auf den Stack geschoben, ist sichergestellt, daß der Stack nicht überlaufen kann, da im ungünstigsten Fall das Intervall halbiert wird. Es hat sich zeitlich jedoch als günstiger erwiesen, diese Unterscheidung nicht zu machen, sondern stets dieselbe Intervallhälfte (hier die rechte) zu speichern. Dadurch ist jedoch die Anzahl der notwendigen Stackelemente nicht mehr begrenzbar, wodurch ein Überlauf eintreten kann. Dieser Spezialfall wird dadurch abzufangen, daß der komplette Sortiervorgang nochmals von vorne gestartet wird.
Abgesehen von diesen Unterschieden kann der Ablauf des Programms analog aus dem Applesoft-Programm übertragen werden.

Kurzhinweise

1. Zweck:
Ampersand-Utility zur Sortierung eines eindimensionalen Arrays nach dem Quicksort-Algorithmus.
2. Konfiguration:
II+, IIe oder IIc;
DOS 3.3
(kein ProDOS wegen HIMEM-Änderung)
3. Test:
RUN QUICKSORT.DEMO
4. Sammeldisk:
QUICKSORT.DEMO
(Applesoft-Demo-Programm)
T.QUICKSORT
(Big-Mac-Quelltext)
QUICKSORT
(Maschinenprogramm)

Ältere Peeker-Hefte

können für DM 6,50 pro Heft zuzüglich Versandspesen angefordert werden. Vergriffene Hefte sind als Photokopien für DM 10,– pro Heft erhältlich.

Dr. A. Hüthig Verlag · Heidelberg

QUICKSORT

BSAVE QUICKSORT, A379Ø7, L493

```
                 1   *-----------------------------------------*
                 2   *                                         *
                 3   *        Ampersand-Utility Quicksort      *
                 4   *                                         *
                 5   *          Harald Grumser, 1985           *
                 6   *-----------------------------------------*
                 7
                 8            ORG  $9413       ;379Ø7
                 9
                 1Ø  START    EQU  $Ø          ;Intervallstart
                 11  LFTPTR   EQU  $2          ;linker Zeiger
                 12  MIDDLE   EQU  $4          ;Intervallmitte
                 13  RGHTPTR  EQU  $6          ;rechter Zeiger
                 14  END      EQU  $8          ;Intervallende
                 15  VALTYP   EQU  $11         ;$FF = String
                 16  INTFLAG  EQU  $12         ;$8Ø = Integer
                 17  SUBFLAG  EQU  $14         ;Integer/DIM-Flag
                 18  YSAV     EQU  $34         ;1. String-Index
                 19  YSAV1    EQU  $35         ;2. String-Index
                 2Ø  DEST     EQU  $6Ø         ;allg. Zeiger
                 21  RESULT   EQU  $62         ;temp. Deskriptor
                 22  CHAR     EQU  $65         ;Vergleichszeichen
                 23  FRETOP   EQU  $6F         ;Ende der Strings
                 24  MEMSIZE  EQU  $73         ;HIMEM
                 25  FAC      EQU  $9D         ;Fließkomma-Akku
                 26  LOWTR    EQU  $9B         ;Arraypointer
                 27  SP       EQU  $EB         ;"Stackpointer"
                 28  LENGHT   EQU  $EC         ;Variablenlänge
                 29
                 3Ø  IN       EQU  $2ØØ        ;Hilf-Stack
                 31  AMPER    EQU  $3F5        ;&-Vektor
                 32
                 33  PTRGET   EQU  $DFE3       ;Var. auswerten
                 34  BSSERR   EQU  $E196       ;"BAD SUBSCRIPT"
                 35  MOVFM    EQU  $EAF9       ;(A,Y) -> FAC
                 36  FCOMP1   EQU  $EBB6       ;(A,Y)/FAC vergl.
                 37
                 38  *            Initialisierung
                 39  *-----------------------------------------*
                 4Ø
9413: A9 94      41  INIT     LDA  #>SORT
9415: AØ 26      42           LDY  #<SORT
9417: 8D F7 Ø3   43           STA  AMPER+2     ;Ampersand,
941A: 8C F6 Ø3   44           STY  AMPER+1
941D: 85 74      45           STA  MEMSIZE+1   ; HIMEM,
941F: 84 73      46           STY  MEMSIZE
9421: 85 7Ø      47           STA  FRETOP+1    ; und FRETOP
9423: 84 6F      48           STY  FRETOP      ; initialisieren
9425: 6Ø         49           RTS
                 5Ø
                 51  *             Hauptprogramm
                 52  *-----------------------------------------*
                 53
                 54  * Array bestimmen und initialisieren
                 55
9426: A9 4Ø      56  SORT     LDA  #%Ø1ØØØØØØ ;Flag für Array-
9428: 85 14      57           STA  SUBFLAG     ; Suche (V-Flag)
942A: 2Ø E3 DF   58           JSR  PTRGET      ;Array auswerten
942D: A9 Ø5      59           LDA  #5          ;Variablenlänge
942F: 24 12      6Ø           BIT  INTFLAG     ; bestimmen:
9431: 1Ø Ø1      61           BPL  NOINT
9433: 4A         62           LSR              ; 2 = Integer
9434: 24 11      63  NOINT    BIT  VALTYP      ; 3 = String
9436: FØ Ø2      64           BEQ  NOSTR       ; 5 = Real
9438: A9 Ø3      65           LDA  #3
943A: 85 EC      66  NOSTR    STA  LENGHT
                 67
943C: AØ Ø4      68           LDY  #$Ø4        ;Zeiger auf DIM-Zahl
943E: B1 9B      69           LDA  (LOWTR),Y
9441: C9 Ø1      7Ø           CMP  #1          ;nur eindimensionale
9442: FØ Ø3      71           BEQ  DIMOK       ; Felder zulässig
9444: 4C 96 E1   72           JMP  BSSERR      ;"BAD SUBSCRIPT"
9447: A5 9B      73  DIMOK    LDA  LOWTR
9449: 69 Ø6      74           ADC  #7-1        ;Schleifenkopf
944B: 85 ØØ      75           STA  START       ;addieren
944D: A5 9C      76           LDA  LOWTR+1
944F: 69 ØØ      77           ADC  #Ø
9451: 85 Ø1      78           STA  START+1
9453: A6 9C      79           LDX  LOWTR+1
9455: AØ Ø2      8Ø           LDY  #$Ø2        ;Feld-Offset
9457: B1 9B      81           LDA  (LOWTR),Y
9459: 65 9B      82           ADC  LOWTR       ;Low-Byte
945B: Ø8         83           PHP              ;Carry der Addition
945C: 38         84           SEC
945D: E5 EC      85           SBC  LENGHT      ;Länge des letzten
945F: 85 Ø8      86           STA  END         ; Elements
9461: BØ Ø1      87           BCS  SP1         ; subtrahieren
9463: CA         88           DEX
9464: 28         89  SP1      PLP              ;Carry aus Addition
9465: C8         9Ø           INY
9466: 8A         91           TXA              ;High-Byte
9467: 71 9B      92           ADC  (LOWTR),Y
9469: 85 Ø9      93           STA  END+1
946B: A9 ØØ      94           LDA  #$ØØ        ;Stack
946D: 85 EB      95           STA  SP          ; initialisieren
946F: FØ 1E      96           BEQ  SUBSORT     ;unbedingt
                 97
                 98  * Neues Intervall vom Stack, falls möglich
                 99
9471: A6 EB      1ØØ  SORTSUB LDX  SP          ;Stackpointer
9473: DØ Ø1      1Ø1          BNE  NOEND       ;falls Null, ...
9475: 6Ø         1Ø2          RTS              ;E N D E
                 1Ø3
9476: BD FC Ø1   1Ø4  NOEND   LDA  IN-4,X      ;neuer
9479: 85 ØØ      1Ø5          STA  START       ; Startwert
947B: BD FD Ø1   1Ø6          LDA  IN-3,X
947E: 85 Ø1      1Ø7          STA  START+1
948Ø: BD FE Ø1   1Ø8          LDA  IN-2,X      ;neuer
9483: 85 Ø8      1Ø9          STA  END         ; Endwert
9485: BD FF Ø1   11Ø          LDA  IN-1,X
9488: 85 Ø9      111          STA  END+1
948A: 8A         112          TXA
948B: E9 Ø4      113          SBC  #4          ;Pointer
948D: 85 EB      114          STA  SP          ; erniedrigen
                 115
                 116  * Zeiger für neues Intervall setzen
                 117
948F: A5 Ø8      118  SUBSORT LDA  END         ;rechten
9491: A6 Ø9      119          LDX  END+1       ; Zeiger
9493: 85 Ø6      12Ø          STA  RGHTPTR     ; initialisieren
9495: 86 Ø7      121          STX  RGHTPTR+1
9497: A5 ØØ      122          LDA  START       ;linken
9499: A6 Ø1      123          LDX  START+1     ; Zeiger
949B: 85 Ø2      124          STA  LFTPTR      ; initialisieren
949D: 86 Ø3      125          STX  LFTPTR+1
949F: 18         126          CLC              ;MIDDLE =
94AØ: 65 Ø8      127          ADC  END         ;(START + END) / 2
94A2: A8         128          TAY              ;bei ungeradzahl.
94A3: A5 ØØ      129          LDA  START       ; Elementen muß
94A5: 45 Ø8      13Ø          EOR  END         ; angepaßt werden
94A7: 24 12      131          BIT  INTFLAG     ;Integer?
94A9: 1Ø Ø4      132          BPL  ADJ         ;nein, dann weiter
94AB: 29 Ø2      133          AND  #%ØØØØØØ1Ø
94AD: DØ Ø4      134          BNE  ADJ1        ;ungeradzahlig
94AF: 29 Ø1      135  ADJ     AND  #%ØØØØØØØ1
94B1: FØ ØA      136          BEQ  NOADJ       ;geradzahlig
94B3: Ø8         137  ADJ1    PHP
94B4: 38         138          SEC              ;ein Element
94B5: 98         139          TYA              ; abziehen, um
94B6: E5 EC      14Ø          SBC  LENGHT      ; geradzahlig
94B8: BØ Ø1      141          BCS  SP2         ; vorzutäuschen
94BA: CA         142          DEX
94BB: 28         143  SP2     PLP
94BC: A8         144          TAY
94BD: 8A         145  NOADJ   TXA
94BE: 65 Ø9      146          ADC  END+1       ;addieren
94CØ: 6A         147          ROR              ; und durch 2
94C1: 85 Ø5      148          STA  MIDDLE+1    ; dividieren
94C3: 98         149          TYA              ; ergibt
94C4: 6A         15Ø          ROR              ; mittleres
94C5: 85 Ø4      151          STA  MIDDLE      ; Element
                 152
                 153  * Mittleren Wert nach FAC übertragen
                 154
94C7: A5 12      155          LDA  INTFLAG     ;Integer?
94C9: 1Ø ØC      156          BPL  FPSTR       ;nein, dann weiter
94CB: AØ Ø2      157  MOVXM   LDY  #2          ;2 oder 3 Bytes
94CD: B1 Ø4      158  MOVLOOP LDA  (MIDDLE),Y  ; für Integer
94CF: 99 9D ØØ   159          STA  FAC,Y       ; oder Strings
94D2: 88         16Ø          DEY              ; übertragen
94D3: 1Ø F8      161          BPL  MOVLOOP
94D5: 3Ø 18      162          BMI  CMPLEFT     ;unbedingt
94D7: A5 11      163  FPSTR   LDA  VALTYP      ;String?
94D9: 3Ø FØ      164          BMI  MOVXM       ;ja, dann weiter
94DB: A5 Ø4      165          LDA  MIDDLE      ;ansonsten
94DD: A4 Ø5      166          LDY  MIDDLE+1    ; mittels
94DF: 2Ø F9 EA   167          JSR  MOVFM       ; ROM-Routine
94E2: 4C EF 94   168          JMP  CMPLEFT
                 169
                 17Ø  * Linken Zeiger erhöhen
                 171
94E5: A5 Ø2      172  INCLEFT LDA  LFTPTR      ;linken
94E7: 65 EC      173          ADC  LENGHT      ; Zeiger um
94E9: 85 Ø2      174          STA  LFTPTR      ; Elementlänge
94EB: 9Ø Ø2      175          BCC  CMPLEFT     ; erhöhen
94ED: E6 Ø3      176          INC  LFTPTR+1
94EF: A5 Ø2      177  CMPLEFT LDA  LFTPTR      ;(A,Y) >=
94F1: A4 Ø3      178          LDY  LFTPTR+1    ; MIDDLE?
94F3: 2Ø 6F 95   179          JSR  COMPARE
94F6: BØ Ø5      18Ø          BCS  CMPRGHT     ;ja, dann weiter
```

```
94F8: 9Ø EB          BCC  INCLEFT     ;weiter erhöhen

                 * Rechten Zeiger erniedrigen

94FA: 2Ø F5 95   DECRGHT  JSR  DECRIGHT
94FD: A5 Ø6      CMPRGHT  LDA  RGHTPTR     ;(A,Y) <=
94FF: A4 Ø7               LDY  RGHTPTR+1   ; MIDDLE?
95Ø1: 2Ø 6F 95            JSR  COMPARE
95Ø4: FØ Ø2               BEQ  SWAP        ;ja, dann weiter
95Ø6: BØ F2               BCS  DECRGHT     ;weiter erniedr.

                 * Vertauschen, falls LFTPTR < RGHTPTR

95Ø8: A5 Ø2      SWAP     LDA  LFTPTR      ;linker Zeiger
95ØA: C5 Ø6               CMP  RGHTPTR     ; größer
95ØC: A5 Ø3               LDA  LFTPTR+1    ; oder gleich
95ØE: E5 Ø7               SBC  RGHTPTR+1   ; rechter Zeiger?
951Ø: BØ 17               BCS  NEXTSUB     ;ja, dann weiter
9512: A4 EC               LDY  LENGHT
9514: 88                  DEY              ;2, 3 oder 5
9515: B1 Ø2      SWPLOOP  LDA  (LFTPTR),Y  ; Bytes
9517: AA                  TAX              ; austauschen
9518: B1 Ø6               LDA  (RGHTPTR),Y
951A: 91 Ø2               STA  (LFTPTR),Y
951C: 8A                  TXA
951D: 91 Ø6               STA  (RGHTPTR),Y
951F: 88                  DEY
952Ø: 1Ø F3               BPL  SWPLOOP
9522: 38                  SEC              ;für Subtraktion
9523: 2Ø F5 95            JSR  DECRIGHT
9526: 18                  CLC              ;für Addition
9527: 9Ø BC               BCC  INCLEFT     ;nächstes Paar

                 * Intervall gegebenenfalls aufteilen

9529: A5 ØØ      NEXTSUB  LDA  START       ;Intervall-
952B: C5 Ø4               CMP  MIDDLE      ; länge
952D: DØ Ø9               BNE  NEXTSUB1    ; 1 oder 2?
952F: A5 Ø1               LDA  START+1
9531: C5 Ø5               CMP  MIDDLE+1
9533: DØ Ø3               BNE  NEXTSUB1    ;nein, dann weiter
9535: 4C 71 94            JMP  SORTSUB     ;R E K U R S I O N

9538: AØ Ø1      NEXTSUB1 LDY  #$Ø1        ;linken und
953A: B9 Ø2 ØØ   NSBLOOP  LDA  LFTPTR,Y    ; rechten
953D: AA                  TAX              ; Zeiger
953E: B9 Ø6 ØØ            LDA  RGHTPTR,Y   ; austauschen
9541: 99 Ø2 ØØ            STA  LFTPTR,Y    ; (falls
9544: 8A                  TXA              ; LFTPTR >
9545: 99 Ø6 ØØ            STA  RGHTPTR,Y   ; RGHTPTR)
9548: 88                  DEY
9549: 1Ø EF               BPL  NSBLOOP

954B: A6 EB               LDX  SP          ;Stackpointer
954D: EØ FC               CPX  #-4         ;Überlauf?
954F: DØ Ø3               BNE  NOOVFL      ;nein, dann weiter
9551: 4C 47 94            JMP  DIMOK       ;nochmal versuchen
9554: AØ ØØ      NOOVFL   LDY  #Ø          ;Zähler
9556: B9 Ø6 ØØ   PUSH     LDA  RGHTPTR,Y
9559: 9D ØØ Ø2            STA  IN,X        ;rechten Zeiger
955C: E8                  INX              ; und Intervall-
955D: C8                  INY              ; ende retten
955E: CØ Ø4               CPY  #4
956Ø: DØ F4               BNE  PUSH
9562: 86 EB               STX  SP          ;neuer Pointer
9564: A5 Ø2               LDA  LFTPTR      ;linker
9566: 85 Ø8               STA  END         ; Zeiger
9568: A5 Ø3               LDA  LFTPTR+1    ; ist neues
956A: 85 Ø9               STA  END+1       ; Intervallende
956C: 4C 8F 94            JMP  SUBSORT     ;R E K U R S I O N

                 *                Unterprogramme
                 *-----------------------------------------------*

                 * 2 Elemente vergleichen (mittleres und (A,Y))

956F: 85 6Ø      COMPARE  STA  DEST        ;Zeiger
9571: 84 61               STY  DEST+1      ; eintragen
9573: A6 12               LDX  INTFLAG     ;Integer?
9575: 1Ø 1A               BPL  COMPFP      ;nein, dann weiter

                 * Integer-Zahlen vergleichen

9577: AØ ØØ      COMPINT  LDY  #$ØØ        ;Int. vergleichen
9579: B1 6Ø               LDA  (DEST),Y
957B: C5 9D               CMP  FAC         ;Zahlen liegen
957D: 9Ø ØA               BCC  CMPINT1
957F: DØ Ø8               BNE  CMPINT1
9581: C8                  INY              ; im High/Low-
9582: B1 6Ø               LDA  (DEST),Y    ; Format vor
9584: C5 9E               CMP  FAC+1
9586: FØ Ø8               BEQ  RTNCPI
9588: 88                  DEY              ;wieder auf MSB
9589: 6A         CMPINT1  ROR              ;C -> Bit 7
958A: 51 6Ø               EOR  (DEST),Y    ;Vorzeichen-Bit
958C: 45 9D               EOR  FAC         ; berücksichtigen
958E: 38                  SEC              ;(löscht Z-Flag)
958F: 2A                  ROL              ;Bit 7 -> C
959Ø: 6Ø         RTNCPI   RTS

                 * Real-Zahlen vergleichen

9591: A6 11      COMPFP   LDX  VALTYP      ;Fließkomma-Zahl?
9593: DØ Ø4               BNE  COMPSTR     ;nein, dann weiter
9595: 38                  SEC              ;(falls FAC = (A,Y))
9596: 4C B6 EB            JMP  FCOMP1      ;FP-Vergleich

                 * Strings vergleichen

9599: AØ Ø2      COMPSTR  LDY  #$Ø2        ;Deskriptor
959B: B1 6Ø      SETDESC  LDA  (DEST),Y    ; als feste
959D: 99 62 ØØ            STA  RESULT,Y    ; Adresse
95AØ: 88                  DEY              ; ablegen
95A1: 1Ø F8               BPL  SETDESC
95A3: 84 34               STY  YSAV        ;initialisieren
95A5: 84 35               STY  YSAV1
95A7: A4 34      CSLOOP   LDY  YSAV        ;1. String-Index
95A9: 24 14               BIT  SUBFLAG     ;V-Flag setzen
95AB: C8         CMPMID   INY              ;nächstes Zeichen
95AC: C4 9D               CPY  FAC         ;Länge
95AE: FØ 3E               BEQ  RTNCS       ; erreicht
95BØ: B1 9E               LDA  (FAC+1),Y   ;MIDDLE-Character
95B2: 7Ø 1Ø               BVS  CHKCHR      ;unbedingt
95B4: 9Ø F5      CMPMID1  BCC  CMPMID      ;ohne Sort.-Wert
95B6: 85 65               STA  CHAR
95B8: 84 34               STY  YSAV
95BA: A4 35               LDY  YSAV1       ;2. String-Index
95BC: B8                  CLV              ;V-Flag löschen
95BD: C8         CMPAY    INY              ;nächstes Zeichen
95BE: C4 62               CPY  RESULT      ;Länge
95CØ: FØ 31               BEQ  RTNCS1      ;fertig
95C2: B1 63               LDA  (RESULT+1),Y ;(A,Y)-Character

95C4: 29 7F      CHKCHR   AND  #%Ø1111111  ;ASCII normieren
95C6: C9 6Ø               CMP  #'a'-1      ;Gemeine durch
95C8: 9Ø Ø2               BCC  VERSAL      ; Versalien
95CA: 29 DF               AND  #%11Ø11111  ; ersetzen
95CC: C9 41      VERSAL   CMP  #'A'        ;alle anderen
95CE: 9Ø 13               BCC  NOSVAL      ; ohne Sort.-Wert
95DØ: C9 5B               CMP  #'Ä'
95D2: DØ Ø2               BNE  NOAE        ;Umlaute
95D4: A9 41               LDA  #'A'        ; durch
95D6: C9 5C      NOAE     CMP  #'Ö'        ; entsprechende
95D8: DØ Ø2               BNE  NOOE        ; Selbstlaute
95DA: A9 4F               LDA  #'O'        ; ersetzen
95DC: C9 5D      NOOE     CMP  #'Ü'
95DE: DØ Ø2               BNE  NOUE
95EØ: A9 55               LDA  #'U'
95E2: 38         NOUE     SEC              ;mit Sort.-Wert
95E3: 7Ø CF      NOSVAL   BVS  CMPMID1     ;V-Flag statt JSR

95E5: 9Ø D6               BCC  CMPAY       ;ohne Sort.-Wert
95E7: 84 35               STY  YSAV1
95E9: C5 65               CMP  CHAR        ;vergleichen
95EB: FØ BA               BEQ  CSLOOP      ;gleich, -> weiter
95ED: 6Ø                  RTS
95EE: A5 62      RTNCS    LDA  RESULT      ;Länge als
95FØ: C5 9D               CMP  FAC         ; Kriterium
95F2: 6Ø                  RTS
95F3: 18         RTNCS1   CLC              ;(A,Y) < FAC
95F4: 6Ø                  RTS

                 * Rechten Zeiger erniedrigen

95F5: A5 Ø6      DECRIGHT LDA  RGHTPTR     ;durch
95F7: E5 EC               SBC  LENGHT      ; Subtraktion
95F9: 85 Ø6               STA  RGHTPTR     ; der
95FB: BØ Ø2               BCS  RTNDR       ; Element-
95FD: C6 Ø7               DEC  RGHTPTR+1   ; länge
95FF: 6Ø         RTNDR    RTS
```

493 Bytes

QUICKSORT.DEMO

```
1Ø PRINT CHR$ (4)"BRUN QUICKSORT"
2Ø DIM A(999)
3Ø FOR I = Ø TO 999
4Ø A(I) = RND (1) * 1ØØØ
5Ø NEXT
6Ø PRINT CHR$ (7): & A: PRINT CHR$ (7)
7Ø FOR I = Ø TO 999
8Ø PRINT A(I): NEXT
```

Vokabeltrainer

Vokabeln lernen mit dem Apple II

von Thomas Zink

Da der Mensch bekanntlich nicht einseitig denken und arbeiten soll, ist es für jeden Apple-Besitzer mindestens genauso wichtig, sich neben den künstlichen Sprachen wie Pascal, BASIC, Assembler usw. auch in natürlichen Sprachen zu üben. Falls Sie eine Fremdsprache erlernen möchten, gehört das Üben der jeweiligen Vokabeln unausweichlich dazu. Nachfolgend wird ein Programm beschrieben, mit dessen Hilfe Sie beliebige Vokabeldateien erstellen und abfragen können.

Wegen des ungewöhnlich großen Umfangs des Programms (allein das Applesoft-Listing würde über 8 Seiten einnehmen) und weil ein Teil der Routinen bereits an anderer Stelle erschienen ist, ist das Programmpaket nebst einer ca. 1000 Vokabeln umfassenden deutsch-englischen Grundwortschatz-Datei nur auf der Peeker-Sammeldiskette enthalten.

1. Allgemeines

Voraussetzung für die Benutzung des Vokabeltrainerprogramms sind ein Minimal-Apple-II oder ein Kompatibler (40 Z/Z ohne Kleinschreibung genügt) mit einem Laufwerk und einer Language-Card, die allerdings nur für das Kopierprogramm benötigt wird. Bei der Benutzung sollte die Taste Shift-Lock eingerastet sein.
Das Programmpaket besteht aus zwei BASIC- und sechs Assemblerprogrammen.

BASIC-Programme:

a. **VOK.TRAINER** – Eigentliches Vokabelprogramm mit 9 Hauptmenü-Optionen (s.u.).
b. **VOK.COPY** – Kleineres BASIC-Programm (über Option 7), das das Kopieren von Disketten steuert.

Assembler-Hilfsprogramme:

a. **CATALOG** – Generiert den für die Optionen 1-6 benötigten speziellen Katalogausdruck.
b. **FREI-SEK** – Berechnet die Anzahl der freien Sektoren auf der im Laufwerk befindlichen Diskette.
c. **GETLINE** – Das normale Applesoft-INPUT interpretiert die Sonderzeichen „““, „,“ und „:“ als Feldtrenner. GETLINE ermöglicht deren Eingabe von der Tastatur und einer Textdatei.
d. **DOS-LOS** – Bereitet eine DOS-lose Datendiskette vor. Ferner wird ein Boot-Programm mit einer Meldung auf Track 0, Sektor 0 geschrieben, damit der Computer beim versehentlichen Booten nicht abstürzt; auch für 40-Spur-Disketten.
e. **RWTS** – Dient für die Programme a, b und d als Read-Write-Track-Sector-Routine, d.h. zum Lesen und Schreiben von beliebigen Sektoren der Diskette.
f. **BCOPY** – (Objekt-Code: VOK.BCOPY) – 1-Drive Kopierprogramm; auch für 40-Spur-Disketten.
Die Routinen FREI-SEK, DOS-LOS, RWTS und BCOPY stammen aus „Apple DOS 3.3“ von U. Stiehl, während die GETLINE-Routine auf H. Grumser zurückgeht (Peeker, 5/85).
Die Programme a-e sind unter dem Namen **VOK.PACK** als Objektcode (von $8F60 bis $9465) zusammengefaßt. Das Kopierprogramm BCOPY wird als einzelne Datei ab $8000 abgelegt. Die Quellcode-Dateien der Maschinenprogramme befinden sich aus Platzgründen nicht auf der Peeker-Sammeldisk.

Einleitend ist vielleicht noch zu erwähnen, daß am Anfang des Vokabeltrainerprogramms der Reset-Vektor auf das Applesoft-Kommando „RUN“ ($D655) gerichtet wird, so daß beim Drücken der Reset-Taste das Programm neu startet. Ferner sei darauf hingewiesen, daß alle BASIC- und Maschinenprogramme, so weit es geht, mit Fehlerabfangroutinen ausgestattet sind, die auf eventuelle Eingabe- oder Diskettenzugriffsfehler (z.B. Input/Output-Error usw.) mit entsprechenden Hinweisen reagieren, um danach das Programm an geeigneter Stelle wieder fortzusetzen.

2. Hauptmenü von VOK.TRAINER

Kurz nach dem Programmstart mit
RUN VOK.TRAINER
erscheint ein Menü, von dem aus Sie 9 Teilprogramme oder Menü-Optionen auswählen können:

(1) Vokabeln abfragen
(2) Vokabeln hinzufügen
(3) Vokabeln berichtigen
(4) Neue Datei eröffnen
(5) Datei löschen
(6) Datei umbenennen
(7) Disketten kopieren
(8) Disketten formatieren
(9) Quit (Ende)

Während sich die ersten drei Optionen (1-3) auf das Abfragen, Hinzufügen und Berichtigen von Vokabeln beziehen, ermöglichen Ihnen die nächsten drei Optionen (4-6) das Eröffnen, Löschen und Umbenennen von Vokabeldateien. Die nächsten beiden (7-8) schließlich gestatten die Manipulation von Disketten, genauer gesagt das Kopieren und Formatieren. Und über Option 9 kann man schließlich das Programm beenden. Danach ist der Reset-Vektor wieder normalisiert.

Die Beantwortung der einzelnen Unteroptionen in den Teilprogrammen erfolgt über die Tastatur. Die erlaubten Tasten sind meistens am Ende des Textes angegeben. Generell ist anzumerken, daß Sie mit
W(unsch)
fast immer zum Hauptmenü zurückgelangen. Von dort aus eröffnet Ihnen die Taste ESC die Einsicht in eine siebenseitige Zusammenstellung der wichtigsten Tastenbelegungen in den einzelnen Teilprogrammen.
Bei den Optionen 1-6 werden Sie zunächst aufgefordert, den Namen der zu bearbeitenden Vokabeldatei einzugeben. In der Mitte des Bildschirms wird ein Inhaltsverzeichnis der bereits vorhandenen Vokabeldateien aufgelistet und darunter die Anzahl der noch freien Sektoren auf der Diskette angezeigt. Überschreitet die Zahl der Dateien 18, so können Sie mit der Leertaste weitere auf der Diskette enthaltene Dateien ausdrucken lassen oder mit Return den weiteren Ausdruck verhindern.

3. Die VOK.TRAINER-Optionen

Die Beschreibung der 8 Hauptmenü-Optionen entspricht nicht dem Bildschirm-Menü, sondern der Reihenfolge der Anlage einer *neuen* Datei.

Option 8: Disketten formatieren

Bevor der Computer Daten auf eine Diskette speichern kann, muß sie bekanntlich formatiert bzw. initialisiert werden. Über Option 8 wird die Diskette nicht wie sonst üblich mit 35 Tracks, sondern mit 40 Tracks (entspricht 20K mehr Speicherplatz) initialisiert (Patch von Wolfgang Schöpe aus mc-Apple-Sonderheft, S. 23). Für diejenigen, die ein Laufwerk besitzen, welches nicht in der Lage ist, den Schreib/Lesekopf über die Breite von 40 Tracks zu bewegen, kann auch mit 35 Tracks formatiert werden. Zur Gewinnung von weiterem Speicherraum wird ferner eine DOS-lose Datendiskette vorbereitet (entspricht 8K mehr Speicherplatz). Insgesamt summiert sich die verfügbare Speicherkapazität der Diskette auf 124K (normal) + 20K + 8K = 152K.

Option 4: Neue Datei eröffnen

Nach der Formatierung der Datendiskette können Vokabeldateien eingerichtet werden (Option 4). Zunächst geben Sie dazu der zukünftigen Datei einen Namen und legen nachfolgend die Sprache fest, in der Sie Vokabeln eingeben und abfragen möchten. Jede so von Ihnen eingerichtete Vokabeldatei wird im Inhaltsverzeichnis aufgelistet und repräsentiert auf der Diskette zwei einzelne Files, einen Binärfile „Name.INFO“ (enthält Informationen über Vokabelanzahl und Fehlerstufen, s.u.) und einen Textfile „Name.VOK“ (enthält die Vokabeln).

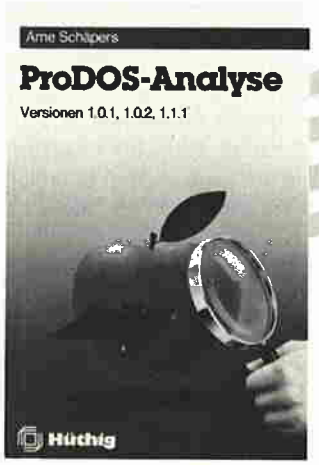
Optionen 5 und 6: Datei löschen und umbenennen

Zur einfacheren Handhabung der Vokabeldateien ermöglichen die Optionen 5 und 6 das gemeinsame Löschen und Umbenennen der Dateien „Name.INFO“ und „Name.VOK“.

Option 2: Vokabeln hinzufügen

Das Abfragen der Vokabeln ergibt natürlich erst dann einen Sinn, wenn Sie zuvor die abzufragenden Vokabeln eingegeben haben. Die Option 2 bietet hierzu die Möglichkeit. Es erscheint eine Bildschirmmaske auf dem Monitor, in der Sie links den Namen der zuvor gewählten Datei und

Computerbücher
die gehen, für Computer die kommen.



Arne Schäpers
ProDOS-Analyse
Versionen 1.0.1, 1.0.2, 1.1.1
1985, 470 S., kart., DM 68,–
ISBN 3-7785-1134-3

Arne Schäpers
Bewegte Apple-Grafik
1985, 305 S., 6 Abb., kart., DM 58,—
ISBN 3-7785-1150-5

Frank Bühler
Applesoft Basic
Tips und Tricks
1985, 241 S., 40 Abb., kart., DM 38,—
ISBN 3-7785-1094-0

Jürgen Kehrel
Assembler lernen
Band 1: Einführung in die Assembler-Programmierung des 6502
1985, ca. 200 S., kart., DM 38,—
ISBN 3-7785-1151-3

Ulrich Stiehl
Apple Assembler
1984, 200 S., 3 Abb., kart., DM 34,—
ISBN 3-7785-1047-9

Ulrich Stiehl
Apple DOS 3.3
Tips und Tricks
3., völlig überarbeitete Ausgabe erscheint Anfang 1986

Ulrich Stiehl
ProDOS für Aufsteiger
Band 1
2., geänderte Auflage 1985, 208 S., kart., DM 28,—
ISBN 3-7785-1098-3

Ulrich Stiehl
ProDOS für Aufsteiger
Band 2
1985, 207 S., kart., DM 30,—
ISBN 3-7785-1036-3

Weitere Titel und Informationen finden Sie in unserem Computerbuch-Katalog:
Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1

Hüthig

AFC COMPUTER

OPERATOR

Qualität und Fortschritt, der sich bewährt hat!

Als Dankeschön für Ihr Vertrauen und die große Nachfrage:

OPERATOR II

jetzt nur noch ..... 595,– DM inkl. MwSt.

Leistungstabelle:

Prozessor:	6511 (ähnl. 6502)
Interface:	parallel/seriell
Handshake wählbar:	ja
Kabel mit Stecker:	ja
Ausführliches Handbuch:	ja
Schaltplan:	ja
Gehäuse:	ergonomisch
Deutsches Erzeugnis:	ja
Hex-Eingabe:	ja
Eingabebuffer:	ja
Barcodeanschluß:	möglich
Passwortprogrammierbar:	ja
Gewicht:	1,9 kg
Maße:	47 × 19 × 3 cm
Programmierbare Tasten:	3×38
Programmierbare Ebenen tauschbar:	alle 3 beliebig
Frei programmierbar:	EEPROM (direkt über Tastaturein-gabe)
Kein Datenverlust nach Abschalten:	ja
Max. Speicherkapazität:	1,6 kByte
Pro Taste:	14 Byte
Vorprogrammierung:	–
Autorepeat:	2 Geschwin-digkeiten
Akustikgeber eingebaut:	ja

Lieferung per Nachnahme zzgl. Versandspesen. Ausführliches Info gegen Freiumschlag.

AFC Computer GmbH
Salmstr. 20 · 5000 Köln 91
Tel. 02 21/83 80 00, Telex 8 873 254 afc

INTUS-Lern- und Anwenderprogramme

für Apple II - Computer

- Rechtschreibtrainer für Deutsch DM 125,–
- Rechtschreibtrainer für Englisch DM 98,–
- Wortschatztrainer Englisch/Deutsch DM 98,– Französisch/Deutsch DM 98,–
- Maschineschreiben wie der Blitz DM 188,–
- Basic-Lernprogramm, sehr umfangreich DM 295,–
- Kinderschule, Lernen für Vorschulkinder DM 59,–
- AppleGraph, Erstellen von Kreis und Balken-Graphiken DM 125,–
- Rechenmodelle für AppleWorks-Rechenblatt DM 195,–
- PriBu-Privatbuchhaltung DM 195,–
- und über 200 weitere Programme. Katalog gratis
- Demo-Disketten mit 5-9 Teilprogrammen DM 10,–
- 6000 Frei-Programme (fast) gratis Programm-Liste (Vorkasse) DM 10,–

Kaiserstr. 21, 7890 Waldshut, Tel. 07751-7920

ccp datentechnik

640 KByte-Drives für den Apple //c!!

- 5¼- od. 3½-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 35/40 Track
- Anschluß an die externe Laufwerkbuchse
- Durch Einbauplatine (kein Löten) 640 KByte im Direktzugriff
- Einfache Anpassung für DOS 3.3, UCSD-Pascal und PRODOS durch menügeführten Patch
- Anpassung von CP/M in Verbindung mit einer Z 80-Zusatzplatine in Vorbereitung
- anschlußfertig im Gehäuse . . . . . **DM 1090,–**

Festplatten für Apple II (//e)

- 5¼ Zoll-Format (Slimline)
- Booten direkt von der Festplatte in DOS 3.3, UCSD-Pascal, PRODOS und CP/M 2.2 / 3.0
- Gemischtbetr. mit 35/40/80/160 Track-Drives
- Copy- und Install-Programme im Lieferumfang
- Umfangreiches Manual
- z. B. 10 MB incl. Netzteil u. Contr., anschlußfertig an Ihren Apple . . . . . **DM 3380,–**

640 KByte-Drives für Apple II (//e)

- 5¼- od. 3½-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 40 Track (Apple kompatible)
- Installationssoftware für DOS 3.3, UCSD-Pascal, CP/M 2.2, CP/M 2.23 (60K), PRODOS, AP22, ALS CP/M+
- Umfangreiches Handbuch
- Anschlußfertige Auslieferung incl. Contr. und 2 Drives
- Diskstation 55II (2 Teac FD55-F, 1.2 MB) . . . **DM 1350,–**
- Diskstation 35II (2 Teac FD35-F, 1.2 MB) . . . **DM 1478,–**

80 Zeichen + 64 K für Apple //e

- und jetzt hinsetzen . . . . . **DM 138,–**

HERDERSTR. 12 2000 HAMBURG 76

☎ 040/ 225676

Anzeigenschluß für Ausgabe 3/86 ist am 17. 1. 86

DIE NEUE DIMENSION

Das modulare 16/32-Bit Computer-System GEPARD

- **CPU MC 68000, 10 M Hz – CPU MC 68020, 16 M Hz**
- **DRAM ab 512 K Byte**
- **Über 30 versch. Steckkarten im Europaformat (100 x 160 mm)**
- **Modula-2 Compiler**
- **Betriebssystem 'PARADISE'**
- **System Editor**
- **CP/M®-68 K**
- **C-Compiler**
- **„Einsteigersysteme" für Apple II und Commodore ab DM 2.598,–**

Wir stellen aus HANNOVER MESSE CeBIT vom 12.3.–19.3.1986 Halle 13, Stand 405

jetzt auch hochauflösende Farbmonitore und schnelle Drucker lieferbar!

Informationsmaterial kostenlos: Tel.: 04 41/7 40 84

GmbH + Co. KG, Westerstr. 10–12, 2900 Oldenburg

Apple II + Kompatible

Komp 48 630,–
48 K. 6502 ohne Firmware

Komp 64
64 K, 6502, Z-80, 15er-Block ohne Firmware 840,–

Komp 64 S 940,–
wie Komp 64, jedoch mit abgesetzter Tastatur mit 188 Funktionen.

Motherboard 48 K 399,–
8 Slots, alle IC's gesockelt, ohne Firmware, fertig geprüft

Motherboard 64 K
wie oben, mit 6502 und Z 80, 64 K 399,–

SUPER PREISE

Klaus Jeschke
Hard-, Software
Viertstr. 3-13
6233 Kelkheim
☎ (0 61 98) 75 23

Alle Preise inklusive Mehrwertsteuer. 6 Monate Garantie Versand erfolgt per NN oder Vorkasse

Für Apple II, IIe

Z-80-Karte	69,–
Disk-Interface	69,–
Centronics-Interf. m. Kabel	69,–
16-K-RAM-Karte	69,–
RS-232-Karte	109,–
Eprommer (4, 8, 16 K)	139,–
128-K-RAM-Karte	279,–
256-KB-RAM-Karte	448,–
Wild-Karte (knackt geschützte Programme)	69,–
80-Zeichen-Karte mit Softswitch, neue Vers. m. gest. scharf. Bild	139,–
Speech-Karte	55,–
Clock-Karte	129,–
Super-Serial-Karte	199,–
Komp 2E Apple 2E kompatible, Rechner 64 K im 2E-Design, ohne Firmware	797,–
80Z + 64K-Karte für 2E kompatible	99,–

Händleranfragen erwünscht!

Apple-Info 1,- DM (Porto)

rechts die Anzahl der bereits eingegebenen Buchstaben (anfangs natürlich null) und die laufende Nummer der Vokabel (1, 2, 3 ...) erkennen können. Der Cursor steht auf der ersten Eingabezeile. Dort geben Sie den ausländischen Teil der Vokabel ein und schließen die Eingabe mit Return ab. Es folgt die Übersetzung (Ende wieder mit Return). Die nun erscheinende Zeile erlaubt Ihnen folgende Möglichkeiten:
– (in Ordnung) J: Abspeicherung der Vokabel
– (in Ordnung) N: Korrektur der Vokabel
– RTN: Beendigung der Eingabe mit Abspeicherung der letzten Vokabel
– E: wie RTN, jedoch ohne Abspeicherung der letzten Vokabel
– S: Sicherung der eingegebenen Vokabelanzahl in der INFO-Datei
Unter S versteht das Programm das Aktualisieren der eingegebenen Vokabelanzahl in der Informationsdatei, das normalerweise nur bei RTN oder E stattfindet. Bei der Eingabe einer größeren Vokabelmenge empfehle ich das Sichern der Vokabelanzahl nach ca. 100 Vokabeln, damit der Verlust bei Ausfall der Anlage gering bleibt. Es bleibt noch zu erwähnen, daß die Länge der gesamten Vokabel 50 Buchstaben und der deutsche oder fremdsprachliche Anteil der Vokabel 35 Buchstaben nicht überschreiten darf. Daraus ergibt sich bei einer DOS-losen und mit 40 Tracks initialisierten Datendiskette eine Speicherungsmöglichkeit von ca. 2850 Vokabeln pro Diskettenseite.

Option 3: Vokabeln berichtigen

Eine andere Leistung des Programms besteht darin, einzelne Vokabeln einer Datei zu berichtigen (Option 3). Voraussetzung dabei ist allerdings die Kenntnis der laufenden Nummer (L) oder des fremdsprachlichen Anteils (F) der Vokabel (also z.B. BOOK für BUCH). Die laufende Nummer steht, wie bereits erwähnt, beim Hinzufügen sowie beim Abfragen in der rechten Ecke des Bildschirms. L muß in Tausender- (T), Hunderter- (H), Zehner- (Z) und Einer- (E) Stellen eingegeben werden. Der Zugriff auf die Diskette erfolgt direkt und der Apple findet die Vokabel in sehr kurzer Zeit. Bei F hingegen muß dem Programm der fremdsprachliche Begriff mitgeteilt und – das ist wichtig – in der gleichen Schreibweise wie der entsprechende Vokabelanteil in der Datei eingegeben werden, d.h. auch die Leerstellen sind von Bedeutung. Dann sucht das Programm von vorne beginnend (bei der 1. Vokabel) die ganze Datei nach der angegebenen Vokabel ab. Dieser Suchvorgang kann bei einer größeren Datei sehr lange dauern. Viel besser ist der Zugriff mit L. Hat Ihr Apple die Vokabel gefunden, liegt es an Ihnen, diese zu berichtigen.

Bei der danach erscheinenden Menüzeile stehen Ihnen vier Möglichkeiten zur Auswahl:
– (In Ordnung) J: Abspeicherung der Korrektur
– (In Ordnund) N: Korrektur der Korrektur
– E: Ende ohne Abspeicherung der Korrektur
– W: zum Menü

Option 1: Vokabeln abfragen

Das eigentliche Lernen der Vokabeln beginnt über Option 1. Es existiert eine ganze Reihe von Möglichkeiten, die Vokabeln abzufragen. Allen gemeinsam ist das Begrenzen des abzufragenden Vokabelbereichs. Nehmen wir einmal an, die Datei enthalte 2000 Vokabeln, die der Lernende nicht kennt. Dann ist es wohl kaum sinnvoll, sofort den ganzen Bereich abzufragen. Aus diesem Grund gestattet das Programm, mit einer unteren (z.B. 500) und einer oberen (z.B. 550) Grenze den zu lernenden Bereich zu limitieren. Weiterhin ist bei allen Abfragemöglichkeiten die Festlegung der Abfragerichtung (z.B. Englisch/Deutsch oder Deutsch/Englisch) obligatorisch.
Nach der Eingabe des Dateinamens und der Festlegung der Grenzen (untere und obere) müssen Sie sich entscheiden, ob der Computer abhängig von einer Fehlerhilfe abfragen soll oder nicht.
Was bedeutet nun die Fehlerhilfe? Die Informationsdatei (Name.INFO) enthält neben der eingegebenen Vokabelanzahl zu jeder Vokabel eine Fehlerstufe (1, 2 oder 3). Unter der Fehlerhilfe versteht das Programm die Benutzung dieser Fehlerstufen beim Abfragen.

Lernen mit/ohne Fehlerhilfe

Bei der Lerntechnik ohne Fehlerhilfe generiert der Apple eine Zufallszahl zwischen der unteren und oberen Grenze des abzufragenden Vokabelbereichs und holt sich die jeweilige Vokabel von der Diskette. Danach wird der deutsche oder fremdsprachliche Anteil der Vokabel (je nach Wahl der Abfragerichtung) auf dem Monitor ausgegeben, und der blinkende Cursor fordert Sie zur Eingabe der Übersetzung auf. Nach Abschluß mit Return stellt der Apple seine und Ihre Übersetzung gegenüber. Mit E(nde) können Sie den Abfragemodus verlassen.

Vor der Erklärung der Abfragetechnik mit Fehlerhilfe soll der Sinn der einzelnen Fehlerstufen näher erläutert werden.

Eine neu hinzugefügte Vokabel besitzt zunächst die Fehlerstufe 1. Wird eine Vokabel abgefragt und Sie wissen die Übersetzung, erhöht das Programm (wenn erwünscht, s.u.) die begleitende Fehlerstufe von 1 auf 2, d.h. alle Vokabeln, die von der Stufe 2 begleitet werden, haben Sie bereits einmal richtig übersetzt. Bei nochmaliger Kenntnis der Übersetzung geht die Fehlerstufe 2 in die Stufe 3 über. Diese Vokabeln (in der Stufe 3) wurden dann von Ihnen mindestens zweimal richtig übersetzt und müssen nur noch von Zeit zu Zeit wiederholt werden. Es kann aber auch vorkommen, daß eine einmalig richtig übersetzte Vokabel (Fehlerstufe 2) wieder in Vergessenheit gerät. In diesem Fall rutscht die begleitende Fehlerstufe von 2 auf 1 zurück. Daraus ergibt sich also folgende Systematik: Die Vokabeln mit der Fehlerstufe
– 1 kennen Sie noch nicht oder sehr schlecht,
– 2 haben Sie mindestens einmal richtig übersetzt,
– 3 haben Sie mindestens zweimal richtig übersetzt.
Ziel ist es natürlich, alle Vokabeln in die Fehlerstufe 3 zu bekommen, um sie somit sicher erlernt zu haben.

Bei der Abfragetechnik mit Fehlerhilfe erwartet das Programm von Ihnen die Angabe, ob die Änderungen der Fehlerstufen in der INFO-Datei (Name.INFO) eingetragen werden sollen oder nicht, d.h. Sie können sich abhängig von den Fehlerstufen abfragen lassen, ohne die neuen Fehlerstufen einzutragen.
Natürlich gibt es auch die Möglichkeit, die Fehlerstufen auf 1 zurückzusetzen (zu löschen), d.h. in ihren Ausgangszustand zu versetzen. Dabei muß man zwischen dem Löschen der Fehlerstufen des ausgewählten, abzufragenden Vokabelbereichs (T = Teilbereich) und dem Löschen der Stufen aller Vokabeln (G = Gesamtbereich) differenzieren.
Schließlich gibt Ihnen der Apple noch die Verteilung der Fehlerstufen in dem zu lernenden Bereich an und fordert Sie auf, die Abfrageart genauer zu bestimmen.
Es werden bei
– 1: nur Vokabeln der Fehlerstufe 1,
– 2: nur Vokabeln der Fehlerstufe 2,
– 3: nur Vokabeln der Fehlerstufe 3,
– A: alle Vokabeln im Fehlerstufenverhältnis 3/2/1,
– R: alle Vokabeln im gewählten Bereich der Reihe nach abgefragt.

Bei A erfolgt das Abfragen bei Gleichverteilung (z.B. bei 30 Vokabeln: 10 Vokabeln der Stufe 1, 10 Vokabeln der Stufe 2 und 10 Vokabeln der Stufe 3) der Fehlerstufen innerhalb der Grenzen im Abfrageverhält-

nis 3 Vokabeln der Stufe 1, 2 Vokabeln der Stufe 2, 1 Vokabel der Stufe 3.
Haben Sie alle Angaben richtig beantwortet und die Abfragerichtung bestimmt, kann das eigentliche Lernen beginnen. Nach der Eingabe Ihrer Übersetzung (oder nur Return, falls Sie den Ausdruck nicht wissen) stellt das Programm seine Übersetzung gegenüber. Sie haben dann drei Möglichkeiten, die nun erscheinende Menüzeile zu beantworten:
– (Zufrieden) J: Sie waren mit Ihrer Antwort zufrieden, und Ihr Apple ändert die entsprechende Fehlerstufe von 1 auf 2, von 2 auf 3 oder die Stufe 3 bleibt erhalten.
– (Zufrieden) N: Sie waren mit Ihrer Antwort nicht zufrieden, und es erfolgt eine Fehlerstufenänderung von 3 auf 2, von 2 auf 1 oder die Stufe 1 bleibt erhalten.
– E: Beendigung des Abfragemodus.

Es bleibt noch zu erwähnen, daß Sie bei Beantwortung der einzelnen Fragen mit S (selbe Parameter) alle Parameter (untere, obere Grenze, Fehlerhilfe, Abfragerichtung usw.) bei der gleichen Datei neu wählen können. Bei der Angabe der unteren oder oberen Grenze des abzufragenden Bereichs bewirkt das Drücken auf die Taste S das Übernehmen der alten, bereits bestimmten Grenzen.

Option 7: Disketten kopieren

Da sich das Eingeben der Vokabeln (Hinzufügen) als sehr arbeitsintensiv erweist, ist es dringend zu empfehlen, von jeder Datendiskette eine Sicherungskopie anzufertigen, damit Sie bei Beschädigung der Originaldiskette auf die Kopie zurückgreifen können. Das Kopieren von Disketten gestattet Ihnen die Hauptmenü-Option 7. Damit für den Kopiervorgang genug Speicherplatz vorhanden ist, wird ein kleines BASIC-Programm mit dem Namen VOK-.COPY geladen. Das eigentliche Kopieren übernimmt allerdings ein Maschinenprogramm mit dem Namen VOK.BCOPY, welches die Language-Card benötigt. Mit dem Drücken auf die Leertaste (Space) leiten Sie einen 40-Track- und mit „3" einen 35-Track-Kopiervorgang ein. Der weitere Dialog erfolgt mit der Leertaste. Es gibt jedoch eine Ausnahme: Ist die Zieldiskette nicht formatiert, können Sie nach dem ersten Einlegen der unbehandelten Diskette mit I(nit) statt Space eine Initialisierung der Disk vor dem Kopiervorgang erreichen. Nach vier Lese- und Schreibvorgängen ist das Duplizieren beendet.

4. Noch einen Tip

Ich empfehle Ihnen, die Vokabeln in 50er-Schritten zu lernen (z.B. Vokabeln 1-50, dann 51-100 usw.). Bei neuen Vokabeln können Sie diese der Reihe nach (Abfrageart R) ohne Fehlereintrag zunächst einmal kennenlernen. Danach fragen Sie nur Vokabeln der Fehlerstufe 1 (am Anfang oder nach dem Löschen sind alle Vokabeln in der Fehlerstufe 1) mit Fehlereintrag solange ab, bis Sie alle Vokabeln einmal richtig übersetzt haben, d.h. alle in der Fehlerstufe 2 enthalten sind. Jetzt ist es ratsam, eine Pause von einem oder mehreren Tagen einzulegen und beim nächsten Lerntermin die Fehlerstufe 2 solange abzufragen, bis alle Vokabeln entweder in die Stufe 1 zurückgefallen (vergessene Vokabeln) oder in die Stufe 3 aufgestiegen sind (zweimal richtig übersetzt). Mit den zurückgefallenen Vokabeln verfahren Sie wie am Anfang. Sie sind also solange zu wiederholen, bis alle wieder in die Stufe 2 aufgerückt sind. Danach sollten Sie eine Pause einlegen (ca. 1 Tag) und darauf nur Vokabeln der Stufe 2 abfragen, bis einige nochmals in die Stufe 1 zurückgefallen und andere in die Stufe 3 aufgestiegen sind. Die Anzahl der in die Stufe 1 zurückgleitenden Vokabeln wird immer kleiner werden. Am Schluß des Lernvorgangs befinden sich (fast) alle Vokabeln in der Fehlerstufe 3.
Natürlich sollten Sie die gelernten Vokabeln von Zeit zu Zeit wieder auffrischen. Wenn Sie Ihre Englischkenntnisse vertiefen oder auffrischen wollen, so können Sie auf die bereits eingegebene Datei zurückgreifen, die sich auf der Peeker-Sammeldiskette befindet.

Kurzhinweise

1. Zweck:
Vokabellernprogramm
2. Konfiguration:
Apple II+/e/c; 40 Z/Z; Kleinschreibung nicht erforderlich; DOS 3.3 (ggf. für 40-Spur-Laufwerke); kein ProDOS!
3. Test:
RUN VOK.TRAINER
dann über Menü-Option 1 (= Vokabeln abfragen) deutsch-englische Übungsdatei „GWS" einlesen.
4. Sammeldisk:
VOK.TRAINER (Hauptprogramm)
VOK.COPY (Kopierprogramm)
VOK.PACK (Trainer-Hilfsroutinen)
VOK.BCOPY (Kopier-Hilfsroutine)
GWS.INFO (Deutsch-Englisch-Info)
GWS.VOK (Deutsch-Englisch-File)

ProDOS-Editor 1.0

Applesoft-Editor unter ProDOS-Betriebssystem

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1024-X

Mit diesem neuen Editor – übrigens der bislang einzige deutsche ProDOS-Editor – wird dem Applesoft-Programmierer ein Werkzeug zur effektiven Programmierung unter dem Betriebssystem ProDOS gegeben, denn die früheren Editoren sind allesamt unter ProDOS nicht mehr lauffähig.

Unter anderem sind folgende Features implementiert worden:

- Zeilenorientierter Editor mit jedem erdenklichen Redigierkomfort (Insert, Delete, Tab, Restore, freie Cursorbewegung in allen vier Richtungen, Eingabe von Ctrl-Buchstaben in Applesoft-Zeilen usw.)
- Renumber (Zeilen-Umnumerierung)
- Xreference (sortierte Variablenliste)
- Suchen von Tokens, Strings und Variablen
- dezimale und hexadezimale Umrechnungen
- Ausführung von Monitorbefehlen aus dem Editor heraus
- Listen des Applesoft-Programms in speicherinterner Form als Hex-Dump
- Suchen von Hex-Folgen, Adressen oder Speicherstellen im gesamten RAM-Bereich einschließlich der Language-Card
- frei definierbare Tastatur-Macrobefehle

Der Applesoft-Editor liegt in einem von ProDOS geschützten Bereich und läßt sich per Tastendruck vorübergehend abschalten und ebenso einfach wieder aktivieren.

Gerätevoraussetzung: Apple II+, IIe oder IIc, 40 Zeichen/Zeile

Hüthig Software Service,
Postfach 10 28 69,
D-6900 Heidelberg

peeker Börse

Verkauf Software

Apple: Public Domain: Pro Volume DM 15,– Games, Schach, Graphic u.v.a.m. Derw. Lehrerprogramme, 'Mini-Logo'! Gratisinfo: Fa. Walt. Muhle, Waldwinkel 3,2105 Seevetal 3

PIRATE DEFENCE 2.0 Kopierschutz Gehört zu den sichersten Schutzsystemen Deutschlands. Für DOS, ProDOS, DIVERSI u. a. Info (50 Pf), Chr. Bregler, Tulpenstr. 2, 7519 Eppingen. Händleranfragen erw.!

MULTIPLAN (Macintosch) DM 290 APPLE ACCESS (IIe/IIc) DM 120 dAdress (dBase II erford.) DM 120 dLager (dBase II erford.) DM 120 U. Blaseg 07529-408

P-Code Disassembler DM 50,–, K. Seiler, Willy-Andreas-Allee 1, 7500 Karlsruhe 1

-Kredit-Programm- für Apple- IIc DM 40,- inkl. Must. Fa. Stampflmeier, 82 Rosenheim, von der Tannstr. 11

Print Shop Newsroom etc. / Apple ändere ich für Ihr Drucker-IF! Disk-Copy für EHRING & ERPHI Super-Prgm. auch PCTEXT/ PC-1500 Rüter Rah. Str. 65, 4955 H. 05703/672

Apple II-Spiele-Programmierung: Hires-Schrift und Sprites (Rout. mit Sourcefiles und Erklärung): Disk nur DM 28,- Tel: 0261/63586

Orgin. Flight Simulator II 100- die verlassene Burg 30,- T: 04101/34327

Verkauf Hardware

128 auf 512K DM 798,– Bausatz DM 298,–, ab 17 h, Telefon 089/985889

Centronic Parallel Grafik Interface, DM 140 VHB, T: 06142/ 603443

Fernschreiberinterface am Gameport m. Programm DM 79,– P. Benner, Hubertusstr. 131, 4150 Krefeld

Apple II komp. mit 80 Zeichen-Pal-V24-Karten u. Paddle User-Buch für DM 700,–, T. 06202/16513

Apple IIc Monit. Maus, Joyst. Imagewriter, Olivetti-Praxis 30 mit Seriellem Interf., 6 Mon. Tel: 02161/26698

NEC P2 incl. V. 24, Autom. Einzelblatteinzug, Kabel für Apple IIc od. MAC, 7 Farbbänder statt NP 3428 nur DM 2308. T: 08806/ 338

Soundchaser Musik-System mit Softw. & Keyboard, Tel: 04743/5500

Apple II + kompatibler IBM Gehäuse 64 KB+Z80 900,- Laufwerk TEAC 55F 400,- ERPHI kon. 250 T: 04101/34327

Matrixdrucker Gemini 10X, Thermodrucker Trendcom 200, 2 Cumana LW, IF (Text-Graph), 80Z, Monitor, kompl. DM 1350,-, Mo-Do 02173/30384, Fr-So 0641/ 33169

Apple II komp. Z80, 80Z, Erphi-Cont., 2×620K Lw, Wordstar-Tastatur, Monitor, Softw., DM 2800 Tel: 0831/95558

Verk. Apple IIc + Monitor +UCSD Pascal + Reference manual Vol. 1 und 2. T: 089/ 8507173 ab 18 h.

Apple II+ orig., 64K, VB DM 2400,- 80Z-Karte 1/2 Jahr für DM 120,- Tel: 06103/34290

Video-Interface zum Einblenden von Schriften mit dem Apple II DM 1075,- bei Ulrich Kallweit, Haus Mallinckrodt, 5802 Wetter 1

Apple IIc + Lit. + ASS.-Tools u. ASS.- Kurs, VB DM 2600, 089/ 6123752

Tastatur, programmierbar f. Apple II im IBM-Look VB. DM 200 T: 0203/471265

Apple IIc, 2.LW, Maas; CP/M-Karte, Software, VB DM 4600; 0221/761845

Verkaufe Drucker-Star mit Graf-Star Karte zus. DM 700,- Tel: 02384/3909

Verschiedenes

APPLE REPARATUREN (auch compatible M-boards, z.B. **Atlas, Arca, CES, Datastar, Dipa, Lasar, Mewa, PC-48 + 64, Plato, Radix, o. ae.) sowie Zusatzkarten und Disk-Drives** führt unser Spezialistenteam mit mehr als 5-jähriger Kunden- und Reparatur-Dienst-Erfahrung, garantiert zuverlässig und besonders kostengünstig aus.
Bitte genaue Fehlerangabe sowie Tel. Nr. für evtl. Rückfragen nicht vergessen.
Auf Wunsch Kostenvoranschlag.
aaa-electronic gmbh
Habsburgerstr. 134, 7800 Freiburg, Tel. 0761/276864, Tx. 772642aaad

Hard- und Software für Apple-Computer gesucht. Wer schreibt gute Software? Auch Gebrauchtes angenehm. Zahle Höchstpreise! Suche noch Leute, die nebenbei verdienen wollen. Zuschriften Chiffre P1005

** Neue Bücher für den Apple II **
Elektronik- und Graphik-Programme
Die Programm-Fundgrube zu folgenden Themen: Statistik, Filter, Netzwerke, Laplace- und Fourier-Transf., Komplexe Rechnung, Diagramme usw., zahlr. Abb., 184 S., 16,5 × 23,5 cm, DM 39,80
Apple II – leicht programmiert
Der Wegweiser zum eigenen Programm: Zählen, Schleifen, Strings, Eingabe, Druckformatierung, Adressverw. usw., 96 S., 16,5 × 23,5 cm, DM 22,80
BASIC-Tricks für den Apple II
Die Trickkiste für die professionelle Programmgestaltung: Eingabe, Menüs, Sortieren, Listengestaltung, Fehlererkennung/Korrektur, Dateien usw., 144 S., 16,5 × 23,5 cm, DM 29,80
Bei Bestellung bitte DM 2,50 für Porto und Verpackung addieren.
beam-Verlag,
Pf. 1148P, 355 Marburg
****

Ankauf Hardware

Günstiges auch gebrauchtes Koala Pad o. ä. für A. II+ T: 02196/2463

Ankauf Software

Datenbank Literaturverwaltung für Apple IIe sucht Telefon 06162/73233

Komfortable Apple Fakturierung ges. für PRODOS, mit Statistik, Gutschrift, Rückstand, usw.

Suche Spiele für Apple T: 02302/89694

Für weitere Informationen zu einem der in dieser Ausgabe vorgestellten Produkte stehen Ihnen die Produktkarten zur Verfügung

Bitte verwenden Sie für Kleinanzeigen die vorgedruckten Antwortkarten in diesem Heft.

Für Ihre Unterlagen

Abonnement bestellt

am: _____

Vertrauensgarantie:
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1 innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

Peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg 1

Abo-Karte

PEEKER

Ja, ich möchte **Peeker** abonnieren.
Liefern Sie mir **Peeker** ab Ausgabe ..... zum Jahresbezugspreis von DM 72,– (Inland) inkl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpakkung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt DM 72,– inkl. MwSt., zzgl. DM 18,– Versandspesen.

Ich wünsche jährliche Berechnung durch:
☐ Verlagsrechnung ☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank / PschA

Bankleitzahl Kto.-Nr.

Datum Unterschrift

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____
_____
_____
_____
_____

am: _____

bei:

Peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

Buch-Karte

PEEKER

Bitte senden Sie mir gegen Rechnung folgende Bücher:

Menge	Autor, Titel	à DM	gesamt DM

Datum Unterschrift

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____
_____
_____
_____
_____

am: _____

bei:

Peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

Software-Karte

PEEKER

Bitte senden Sie mir gegen Rechnung folgende Disketten:

☐ Peeker-Sammeldiskette, einzeln
Disk# _____, Disk# _____
Disk# _____, Disk# _____
Preis je Disk DM 28,– (einzeln)

☐ Peeker-Sammeldiskette,
im Fortsetzungsbezug
ab Disk # _____
(Mindestbezug 6 Disketten)
Preis je Disk DM 20,–
Neben DOS-Disketten auch liefern
☐ CP/M ja ☐ CP/M nein
☐ Pascal ja ☐ Pascal nein

☐ Apple DOS 3.3, Begleitdiskette, DM 28,–
☐ ProDOS, Band 1, Begleitdiskette, DM 28,–
☐ ProDOS, Band 2, Begleitdiskette, DM 28,–
☐ Apple Assembler, Begleitdiskette, DM 28,–
☐ ProDOS-Editor 1.0, Programm, DM 98,–
☐ MMU 2.0, Programm, DM 98,–
☐ INPUT 2.0, Programm, DM 98,–
☐ Softbreaker 1.0, Programm, DM 48,–
☐ DB-Meister, Programm, DM 290,–
☐ Superplot, Programm, DM 48,–
☐ Superquick, Programm, DM 48,–
☐ Turtle Graphics, Programm, DM 98,–

Datum Unterschrift

Abo-Karte

Name

Firma

Abteilung

Straße

PLZ/Ort

Vertrauensgarantie:
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1 innerhalb von 14 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

Datum

Unterschrift

Verlagshinweis:
Das Abonnement verlängert sich zu den jeweils gültigen Bedingungen um ein Jahr, wenn es nicht 2 Monate vor Jahresende schriftlich gekündigt wird.

Buch-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker
Leserservice

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

POSTKARTE

Peeker
Buchabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

POSTKARTE

Peeker
Softwareabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1021-5

„Input 2.0" liegt wahlweise in der Bank 1 oder Bank 2 der Language Card und wird durch einen kurzen Driver in den unteren 48K aufgerufen.
Für jedes Feld der Bildschirmmaske lassen sich u. a. definieren: Feldlänge (bis zu 255 Zeichen) – Vtab – Htab – Datentyp (insgesamt 8 Typen) – Scrollflag (starre oder dynamische Maske) – Ctrlflag – Füllflag – Löschflag – Bildschirmflag (40- oder 80-Z-Darstellung). Innerhalb eines Eingabefeldes besteht jeder denkbare Redigierkomfort (Insert, Delete, Rubout, Restore usw.).
Gerätevoraussetzung: Apple IIe oder IIc; ferner Apple II+ im 40-Zeichenmodus

MMU 2.0

Memory Managements Utilities

für die Apple IIe 64K-Karte DOS 3.3 (und ProDOS)

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7787-1023-1

Insgesamt enthält die neue „MMU 2.0"-Diskette über 25 Programme, die neue Einsatzmöglichkeiten für die Extended 80 Column Card (erweiterte 80-Z-Karte = 64K-Karte für den Apple IIe) erschließen. Ein Teil der Programme laufen auch auf dem Apple II Plus, doch ist „MMU 2.0" primär für 64K-Karte-Besitzer gedacht.

Gerätevoraussetzung: Apple IIe mit 64K-Karte oder IIc

Softbreaker 1.0

Eine softwaremäßige Interrupt-Utility für die Apple IIe 64K-Karte

von U. Stiehl

1984, Diskette und Manual, DM 48,–
ISBN 3-7785-1022-3

Softbreaker ist ein Assemblerprogramm, mit dessen Hilfe Programme, die sich von der 64K-Karte (= Extended 80 Column Card für den Apple IIe) starten lassen, unterbrochen, gespeichert, geladen und exakt an der Stelle der Unterbrechung fortgeführt werden können. Dadurch ist es auch möglich, Sicherungskopien von sogenannten kopiergeschützten Programmen herzustellen.
Mit Softbreaker unterbrochene Programme werden komplett, d. h. die ganzen 64K einschließlich Language Card, in nur ca. 11 Sekunden auf einer formatierten Diskette gesichert.
Gerätevoraussetzung: Apple IIe mit 64K-Karte

Hüthig Software Service, Postfach 10 28 69, D-6900 Heidelberg

Kyan-Pascal

Kyan-Pascal ist ein völlig neuartiges Pascal-System für den Apple II+/e/c, das von der Firma Kyan Software in San Francisco entwickelt wurde und als Version 1.0 im April dieses Jahres erstmals ausgeliefert wurde. Inzwischen gibt es die neueste Version 1.2 vom 14. August 1985.

Vor- und Nachteile von Kyan-Pascal

1. UCSD-Pascal-Implementierungen für den Apple II laufen zwar unter dem 6502-Prozessor und sind deshalb auf den Apple bestmöglich angepaßt, doch wird nur der relativ langsame Pseudo-Code erzeugt. Z80-Pascal-Implementierungen sind demgegenüber zwar in der Regel als Vollcompiler ausgelegt, doch wird erstens eine Z80-Karte benötigt und zweitens läßt die Anpassung an den Apple II, z. B. bezüglich der HGR-Grafik, zu wünschen übrig.

   Kyan-Pascal erzeugt einen echten 6502-Objektcode. Es vereinigt damit die Geschwindigkeitsvorteile eines Vollcompilers mit der bestmöglichen Anpassung an den Apple II.

2. Kyan-Pascal läuft unter allen denkbaren Apple-Konfigurationen (II+/e/c). Man kann es bereits mit Nutzen einsetzen, wenn man nur über eine Minimalkonfiguration mit Apple II+/e ohne 80-Zeichenkarte und ohne zweites Laufwerk verfügt, weil auf der (ungeschützten) Programmdiskette noch über 40.000 Bytes für die Arbeitsdatei frei sind. Wer besser ausgestattet ist, kann eine 80-Zeichenkarte, ein zweites Laufwerk sowie eine RAM-Disk verwenden.

3. Kyan-Pascal läuft unter dem (mitgelieferten) Betriebssystem ProDOS und kann deshalb auch in Verbindung mit größeren Massenspeichern (Festplatten sowie 80-Spur-Disketten usw.) problemlos eingesetzt werden.

4. Im wesentlichen besteht Kyan-Pascal aus einem sehr brauchbaren Fullscreen-Editor (in zwei Versionen für 40- und 80-Zeichendarstellung), der auch als Textverarbeitungsprogramm benutzt werden kann, einem Compiler und einer Runtime-Library (Bibliotheksdatei).

   Fertig compilierte Programme können in Verbindung mit der Bibliotheksdatei unabhängig von Kyan-Pascal als ProDOS-Systemdateien gestartet werden. Die Weitergabe der Runtime-Library wird von der Firma Kyan Software ohne Zahlung von Lizenzgebühr gestattet.

5. Der Kyan-Pascal-Befehlssatz lehnt sich eng an Standard-Pascal an, so daß beispielsweise Befehle wie GOTOXY usw. fehlen. Für String-Verarbeitung und HGR-Grafik werden jedoch Include-Dateien mitgeliefert. Dieser zunächst gravierend erscheinende Nachteil wird jedoch – wie wir meinen – durch zwei entscheidende Vorteile aufgehoben:

   a) Ein laufendes Kyan-Pascal-Programm befindet sich in der von Applesoft her gewohnten Monitor-Umgebung, d. h. der Pascal-READLN-Befehl wird z. B. durch die Monitor-Routine GETLN (entspricht dem Applesoft-INPUT-Befehl) realisiert. Der Aufruf von Monitor- oder gar Applesoft-ROM-Routinen ist deshalb, wenn man auf die Zero-Page Rücksicht nimmt, grundsätzlich möglich.

   b) In einen Kyan-Pascal-Quelltext können beliebig viele echte 6502-Quelltexte eingebaut werden. Man kann deshalb Kyan-Pascal auch als 6502-Assembler benutzen. Ein Beispiel-Programm macht dies deutlicher:

```
PROGRAM DEMO;
BEGIN
  WRITELN ('Jetzt 6502');
#a {Beginn Assembler}
HOME EQU $FC58
BELL EQU $FBDD
  JSR HOME ;Bildschirm löschen
  JSR BELL ;Piepston ausgeben
  {usw.}
#{Ende Assembler}
  WRITELN ('Das war 6502');
END.
```

   Auf diese Weise ist es prinzipiell möglich, beliebige Assemblerroutinen in Kyan-Pascal-Programme einzubauen und darüber hinaus die in Standard-Pascal nicht vorhandenen Befehle zu kompensieren.

6. Kyan-Pascal scheint uns sowohl für Anfänger, insbesondere auch im Schulunterricht, als auch für Fortgeschrittene bestens geeignet zu sein. Anfänger können sich auf den Standard-Befehlssatz beschränken, während Fortgeschrittene die Assembler-Features ausnutzen können. Übrigens erzeugt Kyan-Pascal beim Compilieren ein echtes 6502-Quellcode-Listing, und zwar auch für die Pascal-Befehle, so daß ein nachträgliches Optimieren des Assembler-Quellcodes möglich ist.

Einladung zum Sammelbezug

Kyan-Pascal kostet in den USA 69.95 Dollar. Unter Berücksichtigung der Auslandseinfuhr- und Überweisungskosten muß man umgerechnet über DM 200,– bezahlen. Damit Sie diese interessante Pascal-Implementierung so günstig wie möglich beziehen können, bieten wir Ihnen als Peeker-Serviceleistung die Teilnahme am Sammelbezug an. Als Stückpreis* haben wir DM 170,– (inkl. Porto, Verpackung und MwSt) festgelegt. Voraussetzung hierfür ist allerdings, daß eine ausreichende Anzahl von Vorbestellungen eingeht. Werten Sie bitte diese Aktion als einen Versuch, Software im Sammelbezug preiswert zu erwerben. Sollten sich nicht genügend Interessenten melden, so müssen wir dieses Service-Angebot wieder vergessen. Kyan-Pascal ist freilich auch bei einschlägigen Importeuren (für z. Zt. ca. DM 270,–) erhältlich.

Wie wird's gemacht? Senden Sie uns bis spätestens 31.1.1986 Ihre Bestellung über Kyan-Pascal zum Preis von DM 170,– inkl. Versandspesen. Teilnehmen können alle Peeker-Leser. *Ihre Peeker-Redaktion*

* Diskette + 100seitiges Handbuch

Applesoft-HGR-Erweiterung in Assembler

von Klaus Schäfer

Die Grafik-Befehle des Applesoft-BASIC sind ziemlich spartanisch. Mit dem Maschinenprogramm **AGE** ist es möglich, ein Hires-Bild individuell zu verändern. Das Programm wird über den Ampersand-Vektor angesteuert und ist somit leicht in einem BASIC-Programm einsetzbar. **AGE-.DEMO** zeigt eine Anwendung der neuen Befehle.

1. Die neuen Befehle

Das einfachste neue Kommando ist das &-Zeichen allein. Durch diesen Befehl wird eine Liste der verfügbaren Kommandos sowie deren Syntax ausgegeben:

Verfügbare Kommandos:

TURN <BUF>,<PART>
REFLECT <BUF>,<PART>
INVERT <BUF>,<PART>
SWAP <BUF>,<BUF>
COPY <BUF>,<BUF>
MERGE <BUF>,<BUF>
SHOW <BUF>
SHRINK <BUF>,<PART>
WIDE <BUF>,<PART>

Die Parameter können sowohl als Zahlen als auch als Variablen angegeben werden. **BUF** steht für die Bildschirmseite. Dabei bedeutet 1 = HGR und 2 = HGR2.
PART (im Bereich 0-4) gibt die folgenden Bildschirmausschnitte an:

1 = links oben (Zeile 0-95 und Spalte 0-139);
2 = rechts oben (Zeile 0-95 und Spalte 140-279);
3 = links unten (Zeile 96-191 und Spalte 0-139);
4 = rechts unten (Zeile 96-191 und Spalte 140-279);
0 = der ganze Bildschirm.

Allen folgenden Befehlen muß das &-Zeichen vorangestellt werden.
– **TURN** dreht einen Bildschirmausschnitt um 180 Grad, das Bild wird also auf den Kopf gestellt. Die Parameter sind hierbei BUF, also entweder 1 für HGR oder 2 für HGR2, und PART im Bereich 0-4.
– **REFLECT** spiegelt einen Bildschirmausschnitt (also spiegelverkehrte Darstellung des Bildes), wobei die Parameter die gleichen sind wie bei TURN.
– **INVERT** invertiert einen Bildschirmausschnitt, d.h. es wird das Negativ des Bildes erzeugt: schwarz auf weiß wird weiß auf schwarz. Auch hier sind die Parameter wieder identisch.
Die Befehle SWAP, COPY und MERGE befassen sich jetzt nicht mehr mit einem Bildausschnitt, sondern nur noch mit dem ganzen Bild.
– **SWAP** vertauscht die beiden Hires-Bilder.
– **COPY** kopiert eine Bildschirmseite in die andere.
– **MERGE** verschmelzt beide Bilder zu einem. Der erste Parameter BUF ist hier die Ausgangsseite, also 1 oder 2, der zweite Parameter BUF bezeichnet den Bestimmungsbildschirm.
– **SHOW** ist der einfachste Befehl. Er bewirkt nur, daß die jeweilige Seite gezeigt wird, ohne diese zu löschen (was bei HGR oder HGR2 passieren würde).
Die komplexesten Befehle sind WIDE und SHRINK.
– **SHRINK** verkleinert ein ganzes Hires-Bild zu einem Viertel der Ursprungsgröße. Als Parameter ist hier BUF der Ausgangsbildschirm (also die Seite, in der das zu verkleinernde Bild steht) und PART der Teil des anderen Bildschirms, in den verkleinert werden soll.
– **WIDE** kehrt die Wirkung von SHRINK um, vergrößert also ein Viertel-Bild zu einem ganzen. Parameter hier sind die Ausgangsseite BUF und der Viertel-Ausschnitt PART. Vergrößert wird wieder in die andere Seite.

Beispiele:

&INVERT 2,1 (invertiert HGR2 links oben).
&TURN 1,0 (dreht die ganze HGR).
&COPY 1,2 (kopiert HGR nach HGR2).
&SWAP 1,2 (vertauscht HGR und HGR2).
&MERGE 1,2 (vermischt HGR und HGR2 in die HGR2).
&SHOW 2 (zeigt HGR2).
&WIDE 2,1 (vergrößert HGR2 links oben nach HGR)
&SHRINK 2,4 (verkleinert HGR2 nach HGR rechts unten)

2. Änderungsmöglichkeiten

Das Programm kann recht einfach erweitert werden. Hierzu muß lediglich die Kommandotabelle (Zeile 131-160) sowie die Tabelle der Routinenadressen erweitert und die eigene Routine angehängt werden. Außerdem ist darauf zu achten, daß eventuell die Startadresse des Programms geändert werden muß, da nur noch wenige Bytes bis HIMEM frei sind.
Gekürzt werden kann noch leichter, da einfach das Kommando und die entsprechende Adresse aus der Kommando- und Adreßtabelle gestrichen werden muß. Es genügt dann, die Routine einfach wegzulassen.

3. Beschreibung der Routinen

Das Programm wird unter DOS 3.3 mit
BRUN AGE
gestartet. Die Zeilen 36-49 initialisieren das Programm, d.h. der Ampersand-Vektor wird auf den Programmstart gerichtet

und HIMEM so gesetzt, daß es nicht durch Strings überschrieben wird.
In Zeile 50-117 befindet sich der Command-Interpreter. Hier werden die Kommandos aus der Kommandotabelle (131-160) der Reihe nach mit dem tatsächlich eingegebenen Kommando verglichen. Wenn in Leerzeichen erreicht wird, ist das richtige Kommando gefunden. Das Ende der Tabelle zeigt, daß kein gültiges Kommando gefunden wurde. Falls der Befehl gültig ist, wird die Adresse der Routine aus der Adreßtabelle entnommen und auf dem Stack abgelegt, um dann mit einem RTS angesprungen zu werden.
Von 176-227 steht die Hauptroutine INVERT. Diese besteht aus einer großen Schleife, die die 192 Bildschirmzeilen durchzählt (als Zählregister dient das X-Register). Die Adresse der jeweiligen Zeile steht in COUNT1, im Y-Register werden die 40 Bytes einer Zeile durchgezählt. (Jedes Byte besteht aus 8 Bits, wovon 7 einen Punkt darstellen; 40 * 7 ergibt also unsere 280 Punkte in X-Richtung).
REFLECT (228-300) arbeitet im Prinzip genauso, wobei hier jedoch nicht invertiert wird, sondern der am weitesten rechts stehende Punkt einer Zeile ganz nach links gesetzt wird usw. Dieses Verschieben innerhalb eines Bytes erledigt die Unterroutine ROTATE (556-586).
Bei TURN (301-392) wird ähnlich wie beim Spiegeln innerhalb einer Zeile vertauscht, dann jedoch noch die oberste mit der untersten Zeile vertauscht. Die Zähler laufen wie bei REFLECT.
COPY, SWAP und MERGE sind in einer Routine integriert (414-503). Auch hier werden wieder die 192 Bildschirmzeilen durchgezählt. Entweder wird eine Zeile auf die andere kopiert oder die zwei Zeilen werden vertauscht oder gemeinsam mit EOR verknüpft.
Die folgende Unterroutine CALCCUR wird von fast jeder Hauptroutine benutzt. Hier wird die Adresse des am weitesten links stehenden Bytes einer Zeile (im X-Register) berechnet. Dies geschieht durch die Interpreter-Routine HPOSN.
Noch einfacher ist die nächste Hauptroutine SHOW zu verstehen. Hier werden nur die jeweiligen Softswitches betätigt, um die gewünschte Seite anzuzeigen.
Die Zeilen 587-610 beinhalten die Unterroutine TESTBUF. Hier wird das eingegebene Kommando darauf abgesucht, welche Grafik-Seite angesprochen werden soll. Diese Seite wird dann entsprechend gesetzt (Adresse $00E6 in der Zero Page).
Ab 622 steht die Routine HILFE. Dieser Programmteil gibt die Kommandotabelle auf dem Bildschirm aus.
Die nächste Unterroutine wird von SHRINK verwendet, um zwei Bytes in eines zu verwandeln. Dies geschieht dadurch, daß nur jedes zweite Bit eines Bytes, also nur jeder zweite Punkt, verwendet und hintereinander in ein Byte hineingeschoben wird.
TESTPART ab Zeile 672 fragt das eingegebene Kommando dahingehend ab, welcher Bildschirmausschnitt verlangt wird. Es werden dann in PAR + 1 und PAR + 2 die X- und Y-Koordinaten der linken, oberen Ecke eines Ausschnitts gespeichert.
Die Routine SHRINK ab 707 schließlich verkleinert eine Grafik-Seite auf ein Viertel. Es werden hier zwei geschachtelte Schleifen verwendet, die erste zählt die Zeilen des ganzen Bildes, die zweite die des verkleinerten. SHMER verkleinert dann zwei Bytes zu einem, und da nur jede zweite Zeile des Originalbildes verwendet wird, ergibt sich eine Verkleinerung auf ein Viertel.
WIDE (Zeile 826 ff.) kehrt die Wirkung von SHRINK genau um. Es wird hier ein Byte verdoppelt, indem jedes Bit doppelt gespeichert wird, und jede Zeile des Ausschnitts wird im Ergebnis zweimal verwendet.

Kurzhinweise

1. Zweck:
Ampersand-Erweiterung der Grafik-Befehle in Applesoft.
2. Konfiguration:
Apple II+, IIe oder IIc; DOS 3.3 (kein ProDOS wegen HIMEM-Änderung).
3. Test:
RUN AGE.DEMO
4. Sammeldisk:
AGE.DEMO
(Applesoft-Demo-Programm)
T.AGE
(Big-Mac-Quelltext)
AGE
(Maschinenprogramm).

AGE.DEMO

```
100 REM AGE-Demo
110 REM
120 PRINT CHR$ (4)"BRUN AGE"
130 REM Zeichnen des Bildes
140 PRINT CHR$ (21)
150 HGR : HGR2 : HCOLOR= 3
160 FOR I = 0 TO 279 STEP 4
170 J = I * 96 / 140
180 HPLOT 0,0 TO I,191
190 HPLOT 0,0 TO 279,J
200 NEXT I
210 REM Verkleinern nach HGR
220 & SHOW1
230 FOR I = 1 TO 4: & SHRINK2,I: NEXT I
240 REM Spiegeln und drehen
250 & TURN1,1
260 & REFLECT1,2
270 & TURN1,2
280 & REFLECT1,3
290 & INVERT1,0
300 REM Noch einmal verkleinern
310 FOR I = 1 TO 4
320 & SHOW2
330 & SHRINK1,I
340 NEXT I
350 REM Drehen und wenden
360 & TURN2,0
370 & REFLECT2,0
380 & SHOW1
390 REM Wieder vergrößern
400 & WIDE2,1
410 REM Kommandos zeigen
420 TEXT : HOME
430 &
```

AGE

(nur Quellcode; Objektcode s.u.)

```
1              ORG  $9100
2    *
3    *-----------------------------*
4    *                             *
5    * AGE - Applesoft-Grafik-     *
6    *       Erweiterung           *
7    *                             *
8    * von Klaus Schäfer           *
9    *                             *
10   *-----------------------------*
11   *
12   COMCOUNT EQU  $0
13   Q1       EQU  $0
14   Q2       EQU  $1
15   BYT1     EQU  $2
16   BYT2     EQU  $3
17   BYT3     EQU  $4
18   HBASL    EQU  $26
19   HBASH    EQU  $27
20   HIMEM    EQU  $73
21   TXTPNT   EQU  $B8
22   CHRGET   EQU  $B1
23   PAGE     EQU  $E6
24   COUNT1   EQU  $FA
25   COUNT2   EQU  $FC
26   Z1       EQU  $FE
27   Z2       EQU  $FF
28   PAR      EQU  $300
29   AMPVEC   EQU  $3F5
30   ERROR    EQU  $D412
31   ADDON    EQU  $D998
32   COMBYTE  EQU  $E74C
33   GETBYTE  EQU  $E74F
34   HPOSN    EQU  $F411
35   COUT     EQU  $FDED
36   *
37   * Hier wird die Adresse des
38   * Amper-Interpreters in den
39   * Vektor von Page 3 gepokt.
40   *
41            LDA  #$4C
42            STA  AMPVEC
43            LDA  #<START
44            STA  AMPVEC+1
45            STA  HIMEM
46            LDA  #>START
47            STA  AMPVEC+2
48            STA  HIMEM+1
49            RTS
50   *
51   * Der Amper-Interpreter prüft
52   * die Kommandos auf ihre Syntax
53   *
54   START    BNE  NORMCOM
55            JMP  HILFE
56   NORMCOM  LDX  #$FF
57            LDA  #00
```

```
58              STA  COMCOUNT
59    *
60    * Initialisierung der Pointer
61    *
62    STARTL01 LDY  #$FF
63    STARTL02 INX
64              INY
65              LDA  COTAB1,X
66    *
67    * wenn $20, dann ist das Kommando
68    * gefunden, weil alle Buchstaben
69    * korrekt verglichen wurden.
70    *
71              CMP  #$20
72              BEQ  GEFUNDEN
73    *
74    * Wenn das Ende der Kommando-
75    * tabelle erreicht wird, wurde
76    * das Kommando nicht gefunden.
77    *
78              CMP  #$FF
79              BEQ  SYNERR
80    *
81    * Jetzt wird mit dem eingetippten
82    * Zeichen verglichen, ....
83    *
84              CMP  (TXTPNT),Y
85              BEQ  STARTL02
86    *
87    * wenn es nicht gleich ist, wird
88    * das nächste mögliche Komman-
89    * do verglichen.
90    *
91    STARTL03 INX
92              LDA  COTAB1,X
93              CMP  #$8D
94              BNE  STARTL03
95              INC  COMCOUNT
96              JMP  STARTL01
97    *
98    * Das Kommando wurde gefunden.
99    * Jetzt wird die Adresse aus
100   * einer anderen Tabelle geholt,
101   * um die Routine anzuspringen.
102   *
103   GEFUNDEN LDA  COMCOUNT
104             ASL
105             TAX
106             LDA  COTAB2+1,X
107             PHA
108             LDA  COTAB2,X
109             PHA
110   *
111   * Jetzt wird nur noch das Y-Reg.
112   * zum Textpointer addiert, damit
113   * Applesoft später mit dem
114   * nächsten Kommando fortfährt.
115   *
116             JSR  ADDON
117             RTS
118   *
119   * Diese kleine Routine lädt das
120   * X-Register mit dem Code für
121   * den Syntax-Error und springt in
122   * die Applesoft-Fehlerroutine
123   *
124   SYNERR   LDX  #$10
125             JMP  ERROR
126   *
127   * Diese Tabelle enthält die
128   * Namen der Kommandos, die be-
129   * liebig geändert werden können
130   *
131   HILFTAB1 HEX  8D
132             ASC  "VERFUEGBARE "
133             ASC  "KOMMANDOS:"
134             HEX  8D8D
135   COTAB1   ASC  'TURN <BUF>'
136             ASC  ',<PART>'
137             HEX  8D
138             ASC  'REFLECT <BUF>'
139             ASC  ',<PART>'
140             HEX  8D
141             ASC  'INVERT <BUF>'
142             ASC  ',<PART>'
143             HEX  8D
144             ASC  'SWAP '
145             ASC  '<BUF>,<BUF>'
146             HEX  8D
147             ASC  'COPY '
148             ASC  '<BUF>,<BUF>'
149             HEX  8D
150             ASC  'MERGE '
151             ASC  '<BUF>,<BUF>'
152             HEX  8D
153             ASC  'SHOW <BUF>'
154             HEX  8D
155             ASC  'SHRINK <BUF>'
156             ASC  ',<PART>'
157             HEX  8D
158             ASC  'WIDE <BUF>'
159             ASC  ',<PART>'
160             HEX  8DFF
161   *
162   * Diese Tabelle enthält die
163   * Adressen der einzelnen Rou-
164   * tinen, die durch ein RTS
165   * aufgerufen werden
166   *
167   COTAB2   DA   DREHEN-1
168            DA   MIRROR-1
169            DA   INVERT-1
170            DA   TAUSCH-1
171            DA   KOPIEREN-1
172            DA   MISCHEN-1
173            DA   ZEIGEN-1
174            DA   SHRINK-1
175            DA   WIDE-1
176   *
177   *-----------------------------*
178   *                             *
179   *  1. Grafik-Routine: Invert  *
180   *                             *
181   * Diese Routine invertiert den *
182   * gewünschten Hires-Puffer.   *
183   *                             *
184   *-----------------------------*
185   *
186   * Der gewünschte Puffer-
187   * ausschnitt wird hier bestimmt
188   *
189   INVERT   JSR  TESTBUF
190            JSR  TESTPART
191            JSR  CALCEND
192            LDX  PAR+2
193   *
194   * Jetzt wird die Adresse einer
195   * Bildschirmzeile bestimmt.
196   *
197   INVL01   JSR  CALCCUR
198            LDA  HBASL
199            STA  COUNT1
200            LDA  HBASH
201            STA  COUNT1+1
202   *
203   * Die Bildschirmzeile (40 Bytes
204   * lang) wird hier invertiert:
205   *
206   *     Bsp: #%10010110
207   *     EOR  #%01111111
208   *       =  #%11101001
209   *
210   *
211            LDY  PAR+1
212   INVL02   LDA  (COUNT1),Y
213            EOR  #$7F
214            STA  (COUNT1),Y
215            INY
216   *
217   * Ende einer Zeile?
218   *
219            CPY  PAR+3
220            BNE  INVL02
221            INX
222   *
223   * Ende eines Ausschnitts?
224   *
225            CPX  PAR+4
226            BNE  INVL01
227            RTS
228   *
229   *-----------------------------*
230   *                             *
231   * 2. Grafik-Routine: Spiegel  *
232   *                             *
233   *  Diese Routine spiegelt den *
234   *  gewünschten Hires-Puffer.  *
235   *                             *
236   *-----------------------------*
237   *
238   MIRROR   JSR  TESTBUF
239            JSR  TESTPART
240            JSR  CALCEND
241   *
242   * Bestimmung der Anfangszeile
243   *
244            LDX  PAR+2
245            STX  Z1
246   *
247   * Bestimmung der letzten Zeile
248   *
249            CLC
250            LDA  PAR+3
251            ADC  PAR+1
252            LSR
253            STA  Z2
254   *
255   * Bestimmung der momentanen Zeile
256   *
257   MIRRL01  LDX  Z1
258            JSR  CALCCUR
259            LDA  HBASL
260            STA  COUNT1
261            LDA  HBASH
262            STA  COUNT1+1
263            LDY  PAR+1
264            STY  Q1
265            LDY  PAR+3
266            DEY
267            STY  Q2
268   *
269   * Hier wird jedes Byte gespiegelt
270   * und von rückwärts wieder ein-
271   * gesetzt.
272   *
273   MIRRL02  LDY  Q1
274            LDA  (COUNT1),Y
275            JSR  ROTATE
276            PHA
277            LDY  Q2
278            LDA  (COUNT1),Y
279            JSR  ROTATE
280            LDY  Q1
281            STA  (COUNT1),Y
282            PLA
283            LDY  Q2
284            STA  (COUNT1),Y
285            INC  Q1
286            DEC  Q2
287   *
288   * Zeile fertig?
289   *
290            LDA  Q1
291            CMP  Z2
292            BNE  MIRRL02
293   *
294   * Ausschnitt fertig?
295   *
296            INC  Z1
297            LDA  Z1
298            CMP  PAR+4
299            BNE  MIRRL01
300            RTS
301   *
302   *-----------------------------*
303   *                             *
304   * 3. Grafik-Routine: Drehen   *
305   *                             *
306   * Diese Routine dreht den ge- *
307   * wünschten Hires-Puffer.     *
308   *                             *
309   *-----------------------------*
310   *
311   * Hier wird wieder der Puffer und
312   * der Ausschnitt eingelesen.
313   *
314   DREHEN   JSR  TESTBUF
315            JSR  TESTPART
316            JSR  CALCEND
317   *
318   * In Z1 steht die Anfangszeile,
319   *
320            LDA  PAR+2
321            STA  Z1
322   *
323   * in Z2 die Endzeile
324   *
325            LDX  PAR+4
326            DEX
327            STX  Z2
328            CLC
329            LDA  PAR+4
330            ADC  PAR+2
331            ROR
332            STA  PAR+4
333   *
```

```
334  * In dieser Schleife werden die
335  * Zeilen durchgezählt.
336  *
337  TURNLO1  LDX  Z1
338           JSR  CALCCUR
339           LDA  HBASL
340           STA  COUNT1
341           LDA  HBASH
342           STA  COUNT1+1
343           LDX  Z2
344           JSR  CALCCUR
345           LDA  HBASL
346           STA  COUNT2
347           LDA  HBASH
348           STA  COUNT2+1
349  *
350  * Q1 ist der Zähler für die mo-
351  * mentane Byte-Position in einer
352  * Zeile; Q2 ist das letzte Byte.
353  *
354           LDY  PAR+1
355           STY  Q1
356           LDY  PAR+3
357           DEY
358           STY  Q2
359  *
360  * In dieser Schleife wird jede
361  * Zeile durchgezählt und gedreht
362  *
363  TURNLO2  LDY  Q1
364           LDA  (COUNT1),Y
365           JSR  ROTATE
366           PHA
367           LDY  Q2
368           LDA  (COUNT2),Y
369           JSR  ROTATE
370           LDY  Q1
371           STA  (COUNT1),Y
372           PLA
373           LDY  Q2
374           STA  (COUNT2),Y
375           INC  Q1
376           DEC  Q2
377  *
378  * Ende einer Zeile?
379  *
380           LDA  Q1
381           CMP  PAR+3
382           BNE  TURNLO2
383  *
384           INC  Z1
385           DEC  Z2
386  *
387  * Bildschirm(-ausschnitt) fertig?
388  *
389           LDA  Z1
390           CMP  PAR+4
391           BNE  TURNLO1
392           RTS
393  *
394  *-------------------------*
395  *                         *
396  * Vorentscheidung: kopieren, *
397  * verschmelzen oder tauschen *
398  *                         *
399  *-------------------------*
400  *
401  TAUSCH   LDA  #1
402           BNE  SETFLAG
403  MISCHEN  LDA  #2
404           BNE  SETFLAG
405  KOPIEREN LDA  #0
406  *
407  * In BYT1 wird festgelegt ob
408  * - Tausch, dann 1;
409  * - Mischen, dann 2;
410  * - Kopieren, dann 0.
411  *
412  SETFLAG  STA  BYT1
413           JMP  COPY
414  *
415  *-------------------------*
416  *                         *
417  * 4. Grafik-Routine: bewegt,  *
418  * vertauscht oder verschmelzt *
419  * die beiden Grafik-Seiten.   *
420  *                         *
421  *-------------------------*
422  *
423  * Zuerst wird der Herkunft-Screen
424  * bestimmt und in Q1 abgelegt,
425  * dann der Bestimmung-Screen, der
```

```
426  * in Q2 abgelegt wird.
427  *
428  COPY     JSR  GETBYTE
429           CPX  #1
430           BEQ  COP1
431           CPX  #2
432           BEQ  COP2
433           LDX  #$35
434           JMP  ERROR
435  COP1     LDA  #$20
436           STA  Q1
437           BNE  COP3
438  COP2     LDA  #$40
439           STA  Q1
440  COP3     JSR  COMBYTE
441           CPX  #1
442           BEQ  COP4
443           CPX  #2
444           BEQ  COP5
445           LDX  #$35
446           JMP  ERROR
447  COP4     LDA  #$20
448           STA  Q2
449           BNE  COP6
450  COP5     LDA  #$40
451           STA  Q2
452  COP6     LDX  #0
453           STX  Z1
454  COPY1    LDX  Z1
455  *
456  * Jetzt werden wieder die Basis-
457  * adressen der Herkunfts- und Be-
458  * stimmungszeile festgelegt.
459  *
460           LDA  Q1
461           STA  PAGE
462           JSR  CALCCUR
463           LDA  HBASL
464           STA  COUNT1
465           LDA  HBASH
466           STA  COUNT1+1
467           LDX  Z1
468           LDA  Q2
469           STA  PAGE
470           JSR  CALCCUR
471           LDA  HBASL
472           STA  COUNT2
473           LDA  HBASH
474           STA  COUNT2+1
475  *
476  * Hier wird wieder jede Zeile
477  * entweder kopiert, vertauscht
478  * oder gemischt und im Zielpuffer
479  * abgelegt.
480  *
481           LDY  #0
482  COPY2    LDX  BYT1
483           LDA  (COUNT1),Y
484           CPX  #0
485           BEQ  MOVE
486           CPX  #1
487           BEQ  SWAP
488  MERGE    EOR  (COUNT2),Y
489           SEC
490           BCS  MOVE
491  SWAP     PHA
492           LDA  (COUNT2),Y
493           STA  (COUNT1),Y
494           PLA
495  MOVE     STA  (COUNT2),Y
496           INY
497           CPY  #40
498           BNE  COPY2
499           INC  Z1
500           LDA  Z1
501           CMP  #192
502           BNE  COPY1
503           RTS
504  *
505  *-------------------------*
506  *                         *
507  * Unterroutine: Berechnung  *
508  * der neuen Cursor-Position *
509  *                         *
510  *-------------------------*
511  *
512  * Diese Routine benutzt die Mo-
513  * nitor-HPOSN-Routine, die die
514  * Adresse einer Zeile berechnet
515  * die sich im A-Reg. befindet.
516  *
517  CALCCUR  TXA
```

```
518           PHA
519           LDX  #0
520           LDY  #0
521           JSR  HPOSN
522           PLA
523           TAX
524           RTS
525  *
526  *-------------------------*
527  *                         *
528  * 5. Grafik-Routine: schaltet *
529  * auf Hires-Seite <1> o. <2>. *
530  *                         *
531  *-------------------------*
532  *
533  * HGR oder HGR2?
534  *
535  ZEIGEN   JSR  TESTBUF
536           LDA  PAGE
537           CMP  #$40
538           BEQ  SEITE2
539  *
540  * Ab hier werden nur noch die
541  * Softswitches betätigt:
542  *
543  * - $C050 für Graphik
544  * - $C052 für volle Graphik
545  * - $C057 für Hires-Graphik
546  * - $C054 für 1. Seite
547  * - $C055 für 2. Seite
548  *
549  SEITE1   BIT  $C054
550           JMP  SWITCH
551  SEITE2   BIT  $C055
552  SWITCH   BIT  $C050
553           BIT  $C057
554           BIT  $C052
555           RTS
556  *
557  *-------------------------*
558  *                         *
559  * Unterroutine: spiegelt  *
560  * ein Byte ............ *
561  *                         *
562  *-------------------------*
563  *
564  * Hier wird ein einzelnes Byte
565  * gespiegelt z.B. #$01 zu #$40
566  *
567  ROTATE   LDX  #9
568           ASL
569           BCC  NOCARRY
570           LSR
571  ROTLO    ROL
572           ROR  BYT1
573           DEX
574           BNE  ROTLO
575           LDA  BYT1
576           ORA  #$80
577           RTS
578  *
579  NOCARRY  LSR
580  NCRLO    ROL
581           ROR  BYT1
582           DEX
583           BNE  NCRLO
584           LDA  BYT1
585           AND  #$7F
586           RTS
587  *
588  *-------------------------*
589  *                         *
590  * Unterroutine: testet, welcher *
591  * Puffer verlangt wurde (1/2). *
592  *                         *
593  *-------------------------*
594  *
595  * Hier wird nur der Puffer von
596  * dem Kommando eingelesen.
597  *
598  TESTBUF  JSR  GETBYTE
599           CPX  #1
600           BEQ  BUF1
601           CPX  #2
602           BEQ  BUF2
603           LDX  #$35
604           JMP  ERROR
605  BUF1     LDA  #$20
606           STA  PAGE
607           RTS
608  BUF2     LDA  #$40
609           STA  PAGE
```

```
610          RTS
611  *
612  *-----------------------------*
613  *                             *
614  * 6. Routine: Hilfe - gibt die *
615  * möglichen Kommandos aus.     *
616  *                             *
617  *-----------------------------*
618  *
619  * Diese Routine benutzt die
620  * normale Kommandotabelle, so
621  * daß jede Änderung eines Kom-
622  * mandonamens richtig angezeigt
623  * wird (z.B. TURN -> DREHEN)
624  *
625  HILFE    LDA  #<HILFTAB1
626           STA  COUNT1
627           LDA  #>HILFTAB1
628           STA  COUNT1+1
629           LDY  #0
630  HILFL01  LDA  (COUNT1),Y
631           CMP  #$FF
632           BEQ  HILFEX1
633           ORA  #$80
634           INY
635           JSR  COUT
636           BNE  HILFL01
637  *
638  HILFEX1  RTS
639  *
640  *-----------------------------*
641  *                             *
642  *  Unterroutine: verschmelzt  *
643  *    zwei Bytes zu einem!     *
644  *                             *
645  *-----------------------------*
646  *
647  * Hier wird nur jedes zweite Bit
648  * eines Bytes erfaßt, so daß
649  * man zwei in einem zusammenfas-
650  * sen kann.
651  *
652  SHMER    LDX  #4
653  SHMERL0  ROL
654           ROL  BYT1
655           ROL
656           DEX
657           BNE  SHMERL0
658           RTS
659  *
660  *-----------------------------*
661  *                             *
662  * Unterroutine: Diese Routine *
663  *  testet, welcher Teil eines *
664  *    Puffers verlangt wird.   *
665  *                             *
666  *-----------------------------*
667  *
668  * Das ist recht einfach, da nur
669  * die passenden X- und Y-Werte
670  * gesetzt werden müssen.
671  *
672  TESTPART JSR  COMBYTE
673           STX  PAR
674           CPX  #4
675           BEQ  VIER
676           CPX  #3
677           BEQ  DREI
678           CPX  #2
679           BEQ  ZWEI
680           CPX  #1
681           BEQ  EINS
682           CPX  #0
683           BEQ  NULL
684           LDX  #$35
685           JMP  ERROR
686  NULL     NOP
687  EINS     LDA  #00
688           STA  PAR+1
689           LDA  #00
690           STA  PAR+2
691           BEQ  TPEX
692  ZWEI     LDA  #20
693           STA  PAR+1
694           LDA  #0
695           STA  PAR+2
696           BEQ  TPEX
697  DREI     LDA  #00
698           STA  PAR+1
699           LDA  #96
700           STA  PAR+2
701           BNE  TPEX
```

```
702  VIER     LDA  #20
703           STA  PAR+1
704           LDA  #96
705           STA  PAR+2
706  TPEX     RTS
707  *
708  *-----------------------------*
709  *                             *
710  *  7. Grafik-Routine: Shrink  *
711  *  Diese Routine verkleinert  *
712  *  einen ganzen Bildschirm zu *
713  *  einem Viertel-Bildschirm.  *
714  *                             *
715  *-----------------------------*
716  *
717  * Es wird wieder der Herkunfts-
718  * puffer sowie der Bestimmungs-
719  * 1/4-Puffer festgestellt.
720  *
721  SHRINK   JSR  TESTBUF
722           JSR  TESTPART
723           LDX  #0
724           STX  Z1
725           LDX  PAR+2
726           STX  Z2
727           LDA  PAGE
728           EOR  #$60
729           STA  PAGE
730  *
731  * Wieder die Feststellung der
732  * Anfangsadresse .....
733  *
734  SHRINK1  LDA  PAGE
735           EOR  #$60
736           STA  PAGE
737           LDX  Z1
738           JSR  CALCCUR
739           LDA  HBASL
740           STA  COUNT1
741           LDA  HBASH
742           STA  COUNT1+1
743           LDA  PAGE
744           EOR  #$60
745           STA  PAGE
746  *
747  * sowie der Bestimmungsadresse
748  *
749           LDX  Z2
750           JSR  CALCCUR
751           LDA  HBASL
752           STA  COUNT2
753           LDA  HBASH
754           STA  COUNT2+1
755           LDY  #00
756           STY  Q1
757           LDY  PAR+1
758           STY  Q2
759  *
760  * In dieser Schleife wird jedes
761  * Byte zusammen mit einem anderen
762  * zu einem zusammengepreßt, und
763  * dann in den Zielpuffer
764  * übertragen.
765  *
766  SHRINK2  LDY  Q1
767           INY
768           LDA  (COUNT1),Y
769           JSR  SHMER
770           DEY
771           LDA  (COUNT1),Y
772           ROL
773           JSR  SHMER
774           LDY  Q2
775           LDA  BYT1
776           STA  (COUNT2),Y
777           INC  Q1
778           INC  Q1
779           INC  Q2
780  *
781  * Ende einer Zeile?
782  *
783           LDA  Q1
784           CMP  #40
785           BNE  SHRINK2
786           INC  Z1
787           INC  Z1
788           INC  Z2
789  *
790  * Ende des Screens?
791  *
792           LDA  Z1
793           CMP  #192
```

```
794           BNE  SHRINK1
795           RTS
796  *
797  *-----------------------------*
798  *                             *
799  *   Unterroutine: Calcend     *
800  *  Berechnet das Ende eines   *
801  *       Viertel-Screens.      *
802  *                             *
803  *-----------------------------*
804  *
805  * In dieser Unterroutine werden
806  * die X- und die Y-Koordinate
807  * der rechten unteren Ecke eines
808  * 1/4-Puffers festgestellt.
809  *
810  CALCEND  LDA  PAR
811           BEQ  GANZ
812           CLC
813           LDA  PAR+1
814           ADC  #20
815           STA  PAR+3
816           CLC
817           LDA  PAR+2
818           ADC  #96
819           STA  PAR+4
820           RTS
821  GANZ     LDA  #40
822           STA  PAR+3
823           LDA  #192
824           STA  PAR+4
825           RTS
826  *
827  *-----------------------------*
828  *                             *
829  * 7. Grafik-Routine: Hier wird *
830  *  1/4-Puffer zu einem ganzen *
831  *      Screen vergrößert.     *
832  *                             *
833  *-----------------------------*
834  *
835  * TESTBUF und TESTPART stellen
836  * wieder den Herkunft-1/4-Screen
837  * und den Bestimmung-Screen fest.
838  *
839  WIDE     JSR  TESTBUF
840           JSR  TESTPART
841  *
842  * Wieder Anfangszeile/Endzeile
843  *
844           LDX  PAR+2
845           STX  Z1
846           LDX  #0
847           STX  Z2
848           LDX  #0
849           STX  BYT3
850  *
851  * Hier werden wieder die Zeilen
852  * durchgezählt.
853  *
854  WIDE1    LDX  Z1
855           JSR  CALCCUR
856           LDA  HBASL
857           STA  COUNT1
858           LDA  HBASH
859           STA  COUNT1+1
860           LDA  PAGE
861           EOR  #$60
862           STA  PAGE
863           LDX  Z2
864           JSR  CALCCUR
865           LDA  HBASL
866           STA  COUNT2
867           LDA  HBASH
868           STA  COUNT2+1
869           LDA  PAGE
870           EOR  #$60
871           STA  PAGE
872           LDY  PAR+1
873           STY  Q1
874           LDY  #0
875           STY  Q2
876  *
877  * In dieser Schleife wird jede
878  * Zeile verdoppelt sowie jedes
879  * Byte ebenfalls bitweise verdop-
880  * pelt (z.B #$01 ->#$03)
881  *
882  WIDE2    LDY  Q1
883           LDA  (COUNT1),Y
884           JSR  WIDEIT
885           LDY  Q2
```

```
886             INY
887             LDA  BYT1
888             STA  (COUNT2),Y
889             DEY
890             LDA  BYT2
891             STA  (COUNT2),Y
892             INC  Q1
893             INC  Q2
894             INC  Q2
895   *
896   * Ende der Zeile?
897   *
898             LDA  Q2
899             CMP  #40
900             BNE  WIDE2
901             LDA  BYT3
902             BEQ  NOINC
903             INC  Z1
904   NOINC     LDA  BYT3
905             EOR  #$01
906             STA  BYT3
907             INC  Z2
908   *
909   * Ende des Screens?
910   *
911             LDA  Z2
912             CMP  #192
913             BNE  WIDE1
914             RTS
915   *
916   WIDEIT    JSR  WDITSUB2
917             LDX  #0
918             JSR  WDITSUB1
919             JSR  WDITSUB2
920             LDX  #1
921             JSR  WDITSUB1
922             RTS
923   *
924   WDITSUB1  LDY  #3
925   WDITLO1   ASL
926             PHP
927             ROL  BYT1,X
928             PLP
929             ROL  BYT1,X
930             DEY
931             BNE  WDITLO1
932             RTS
933   *
934   WDITSUB2  ASL
935             PHP
936             ROL  BYT1
937             PLP
938             ROL  BYT2
939             RTS
```

Der nächste Peeker Heft 2/1986 erscheint am 20. 01. 1986

AGE

(Objektcode)

```
BSAVE AGE, A$9100, L$04B0

$9100: A9 4C 8D F5 03 A9 14 8D
$9108: F6 03 85 73 A9 91 8D F7
$9110: 03 85 74 60 D0 03 4C 0D
$9118: 94 A2 FF A9 00 85 00 A0
$9120: FF E8 C8 BD 6D 91 C9 20
$9128: F0 15 C9 FF F0 21 D1 B8
$9130: F0 EF E8 BD 6D 91 C9 8D
$9138: D0 F8 E6 00 4C 1F 91 A5
$9140: 00 0A AA BD 0F 92 48 BD
$9148: 0E 92 48 20 98 D9 60 A2
$9150: 10 4C 12 D4 8D D6 C5 D2
$9158: C6 D5 C5 C7 C2 C1 D2 C5
$9160: A0 CB CF CD CD C1 CE C4
$9168: CF D3 BA 8D 8D 54 55 52
$9170: 4E 20 3C 42 55 46 3E 2C
$9178: 3C 50 41 52 54 3E 8D 52
$9180: 45 46 4C 45 43 54 20 3C
$9188: 42 55 46 3E 2C 3C 50 41
$9190: 52 54 3E 8D 49 4E 56 45
$9198: 52 54 20 3C 42 55 46 3E
$91A0: 2C 3C 50 41 52 54 3E 8D
$91A8: 53 57 41 50 20 3C 42 55
$91B0: 46 3E 2C 3C 42 55 46 3E
$91B8: 8D 43 4F 50 59 20 3C 42
$91C0: 55 46 3E 2C 3C 42 55 46
$91C8: 3E 8D 4D 45 52 47 45 20
$91D0: 3C 42 55 46 3E 2C 3C 42
$91D8: 55 46 3E 8D 53 48 4F 57
$91E0: 20 3C 42 55 46 3E 8D 53
$91E8: 48 52 49 4E 4B 20 3C 42
$91F0: 55 46 3E 2C 3C 50 41 52
$91F8: 54 3E 8D 57 49 44 45 20
$9200: 3C 42 55 46 3E 2C 3C 50
$9208: 41 52 54 3E 8D FF A8 92
$9210: 4C 92 1F 92 1B 93 23 93
$9218: 1F 93 B9 93 7E 94 13 95
$9220: 20 F3 93 20 30 94 20 F1
$9228: 94 AE 02 03 20 AE 93 A5
$9230: 26 85 FA A5 27 85 FB AC
$9238: 01 03 B1 FA 49 7F 91 FA
$9240: C8 CC 03 03 D0 F4 E8 EC
$9248: 04 03 D0 E0 60 20 F3 93
$9250: 20 30 94 20 F1 94 AE 02
$9258: 03 86 FE 18 AD 03 03 6D
$9260: 01 03 4A 85 FF A6 FE 20
$9268: AE 93 A5 26 85 FA A5 27
$9270: 85 FB AC 01 03 84 00 AC
$9278: 03 03 88 84 01 A4 00 B1
$9280: FA 20 D6 93 48 A4 01 B1
$9288: FA 20 D6 93 A4 00 91 FA
$9290: 68 A4 01 91 FA E6 00 C6
$9298: 01 A5 00 C5 FF D0 DE E6
$92A0: FE A5 FE CD 04 03 D0 BD
$92A8: 60 20 F3 93 20 30 94 20
$92B0: F1 94 AD 02 03 85 FE AE
$92B8: 04 03 CA 86 FF 18 AD 04
$92C0: 03 6D 02 03 6A 8D 04 03
$92C8: A6 FE 20 AE 93 A5 26 85
$92D0: FA A5 27 85 FB A6 FF 20
$92D8: AE 93 A5 26 85 FC A5 27
$92E0: 85 FD AC 01 03 84 00 AC
$92E8: 03 03 88 84 01 A4 00 B1
$92F0: FA 20 D6 93 48 A4 01 B1
$92F8: FC 20 D6 93 A4 00 91 FA
$9300: 68 A4 01 91 FC E6 00 C6
$9308: 01 A5 00 CD 03 03 D0 DD
$9310: E6 FE C6 FF A5 FE CD 04
$9318: 03 D0 AD 60 A9 01 D0 06
$9320: A9 02 D0 02 A9 00 85 02
$9328: 4C 2B 93 20 4F E7 E0 01
$9330: F0 09 E0 02 F0 0B A2 35
$9338: 4C 12 D4 A9 20 85 00 D0
$9378: 85 E6 20 AE 93 A5 26 85
$9380: FC A5 27 85 FD A0 00 A6
$9388: 02 B1 FA E0 00 F0 0F E0
$9390: 01 F0 05 51 FC 38 B0 06
$9398: 48 B1 FC 91 FA 68 91 FC
$93A0: C8 C0 28 D0 E2 E6 FE A5
$93A8: FE C9 C0 D0 B6 60 8A 48
$93B0: A2 00 A0 00 20 11 F4 68
$93B8: AA 60 20 F3 93 A5 E6 C9
$93C0: 40 F0 06 2C 54 C0 4C CC
$93C8: 93 2C 55 C0 2C 50 C0 2C
$93D0: 57 C0 2C 52 C0 60 A2 09
$93D8: 0A 90 0C 4A 2A 66 02 CA
$93E0: D0 FA A5 02 09 80 60 4A
$93E8: 2A 66 02 CA D0 FA A5 02
$93F0: 29 7F 60 20 4F E7 E0 01
$93F8: F0 09 E0 02 F0 0A A2 35
$9400: 4C 12 D4 A9 20 85 E6 60
$9408: A9 40 85 E6 60 A9 54 85
$9410: FA A9 91 85 FB A0 00 B1
$9418: FA C9 FF F0 08 09 80 C8
$9420: 20 ED FD D0 F2 60 A2 04
$9428: 2A 26 02 2A CA D0 F9 60
$9430: 20 4C E7 8E 00 03 E0 04
$9438: F0 3A E0 03 F0 2A E0 02
$9440: F0 1A E0 01 F0 0A E0 00
$9448: F0 05 A2 35 4C 12 D4 EA
$9450: A9 00 8D 01 03 A9 00 8D
$9458: 02 03 F0 22 A9 14 8D 01
$9460: 03 A9 00 8D 02 03 F0 16
$9468: A9 00 8D 01 03 A9 60 8D
$9470: 02 03 D0 0A A9 14 8D 01
$9478: 03 A9 60 8D 02 03 60 20
$9480: F3 93 20 30 94 A2 00 86
$9488: FE AE 02 03 86 FF A5 E6
$9490: 49 60 85 E6 A5 E6 49 60
$9498: 85 E6 A6 FE 20 AE 93 A5
$94A0: 26 85 FA A5 27 85 FB A5
$94A8: E6 49 60 85 E6 A6 FF 20
$94B0: AE 93 A5 26 85 FC A5 27
$94B8: 85 FD A0 00 84 00 AC 01
$94C0: 03 84 01 A4 00 C8 B1 FA
$94C8: 20 26 94 88 B1 FA 2A 20
$94D0: 26 94 A4 01 A5 02 91 FC
$94D8: E6 00 E6 00 E6 01 A5 00
$94E0: C9 28 D0 DF E6 FE E6 FE
$94E8: E6 FF A5 FE C9 C0 D0 A4
$94F0: 60 AD 00 03 F0 13 18 AD
$94F8: 01 03 69 14 8D 03 03 18
$9500: AD 02 03 69 60 8D 04 03
$9508: 60 A9 28 8D 03 03 A9 C0
$9510: 8D 04 03 60 20 F3 93 20
$9518: 30 94 AE 02 03 86 FE A2
$9520: 00 86 FF A2 00 86 04 A6
$9528: FE 20 AE 93 A5 26 85 FA
$9530: A5 27 85 FB A5 E6 49 60
$9538: 85 E6 A6 FF 20 AE 93 A5
$9540: 26 85 FC A5 27 85 FD A5
$9548: E6 49 60 85 E6 AC 01 03
$9550: 84 00 A0 00 84 01 A4 00
$9558: B1 FA 20 8A 95 A4 01 C8
$9560: A5 02 91 FC 88 A5 03 91
$9568: FC E6 00 E6 01 E6 01 A5
$9570: 01 C9 28 D0 E1 A5 04 F0
$9578: 02 E6 FE A5 04 49 01 85
$9580: 04 E6 FF A5 FF C9 C0 D0
$9588: 9E 60 20 A8 95 A2 00 20
$9590: 9B 95 20 A8 95 A2 01 20
$9598: 9B 95 60 A0 03 0A 08 36
$95A0: 02 28 36 02 88 D0 F6 60
$95A8: 0A 08 26 02 28 26 03 60
$9340: 04 A9 40 85 00 20 4C E7
$9348: E0 01 F0 09 E0 02 F0 0B
$9350: A2 35 4C 12 D4 A9 20 85
$9358: 01 D0 04 A9 40 85 01 A2
$9360: 00 86 FE A6 FE A5 00 85
$9368: E6 20 AE 93 A5 26 85 FA
$9370: A5 27 85 FB A6 FE A5 01
```

PROTODOS

Konvertierung von ProDOS- in DOS-3.3-Dateien

von Arne Schäpers

1. Programmbeschreibung

Das Assemblerprogramm **PROTODOS**, das unter DOS 3.3 gestartet wird, kopiert Dateien von einer ProDOS-Diskette auf eine DOS-3.3-Diskette. Es setzt 2 Laufwerke voraus, die an demselben Slot angeschlossen sein müssen, d.h. eine Kopie von z.B. S4, D2 nach S6, D1 ist nicht möglich.
Die zu übertragenden Dateien müssen sich im Volume-Directory (= Hauptinhaltsverzeichnis) der ProDOS-Diskette befinden. Diese Beschränkung erfolgt aus Platzgründen: Eine Version, die zusätzlich das Präfix setzt und auf Subdirectories zugreift, wäre zu umfangreich für eine Veröffentlichung im Peeker.
Das Gegenstück zu PROTODOS, nämlich DOSTOPRO, steht in „Apple ProDOS für Aufsteiger", Bd. 2, S. 183ff. (Hüthig-Verlag).

Die folgenden **Dateitypen** werden von dem Programm unterstützt:

– Applesoft-Programme: ProDOS-Dateityp BAS wird zum DOS-Dateityp A;
– Binärdateien: ProDOS-Dateityp BIN wird zum DOS-Dateityp B, die Startadresse bleibt unverändert;
– Systemdateien: ProDOS-Dateityp SYS wird zum DOS-Dateityp B und erhält die Startadresse $2000;
– Textdateien: ProDOS-Dateityp TXT wird zum DOS-Dateityp T.

Sequentielle Textdateien werden (bis auf das DOS-gerechte Setzen des höchstwertigen Bit 7) unverändert und in beliebiger Länge übertragen.
Was *Random-Textdateien* betrifft, hat PROTODOS eine Neuerung zu bieten: Dieser Dateityp kann ebenfalls übertragen werden, denn das Programm bricht nicht beim ersten leeren Record ab. Statt dessen werden Random-Textdateien expandiert, d.h. unbelegte ProDOS-Records erscheinen auf der DOS-Diskette als leere Sektoren.

2. Bedienungsanleitung

Die „Benutzeroberfläche" von PROTODOS orientiert sich sehr stark an dem bereits im Peeker, Heft 3/85, erschienenen Programm WS.TRANSFER von Dr. Jürgen Kehrel. Die notwendigen Schritte:

– DOS 3.3 von Laufwerk 1 booten;
– PROTODOS mit BRUN PROTODOS starten;
– Die ProDOS-Diskette in Laufwerk 2 einlegen.

Zunächst erhalten Sie einen CATALOG des Volume-Directory der ProDOS-Diskette. Übertragbare Dateien sind dabei bereits mit dem entsprechenden DOS-Dateityp gekennzeichnet. Als nächstes müssen Sie den Namen der zu übertragenden ProDOS-Datei sowie den Namen, den diese Datei auf der DOS-Diskette haben soll, eingeben. Wenn Sie anstelle eines dieser beiden Namen nur ein <Return> eingeben, beginnt das Programm von vorne.
Nach einer Überprüfung, ob nicht bereits eine Datei mit dem angegebenen Namen auf der DOS-Diskette existiert (und ggf. entsprechender Rückfrage des Programms beim Benutzer), beginnt der Übertragungsvorgang. Wenn die Übertragung fehlerfrei abgeschlossen wurde, erhalten Sie danach einen CATALOG der DOS-Diskette.

3. Grenzen des Programms

PROTODOS benutzt diejenige Slotnummer, die beim Programmstart von DOS 3.3 gesetzt wird. Damit wird eine zusätzliche Anfrage an den Benutzer umgangen. Der Transfer zwischen zwei verschiedenen Slots ist daduch allerdings nicht mehr möglich.
Das Speichern eines Binärprogramms mit einer Länge von mehr als $7FFF Bytes ist unter DOS 3.3 ohne vorherigen DOS-Patch nicht möglich und ergibt einen RANGE ERROR. Erstaunlicherweise werden aber von PROTODOS geschriebene Binärdateien von DOS auch dann ohne Probleme geladen, wenn sie länger als $8000 Bytes sind.
Die Expansion einer Random-Textdatei beansprucht u.U. viel Speicherplatz auf der Diskette. Eine Datei mit einer Recordlänge von 100 Bytes und einem einzigen tatsächlich belegten Record mit der Nummer 1300 füllt eine normale DOS-Diskette (35 Spuren) vollständig.
Hier wäre ein Ansatzpunkt für eine Erweiterung des Programms: In dem sog. Auxiliary-Information-Feld (AUX_INFO) des ProDOS-Dateieintrags findet sich bei einer Random-Textdatei die Recordlänge, mit der die entsprechende DOS-Datei über den File-Manager eröffnet werden könnte. Das Programm müßte beim Lesen der ProDOS-Datei danach jeden einzelnen Record prüfen und nicht-existente Records auf der DOS-Diskette mit Hilfe des File-Manager-POSITION-Befehls überspringen. Allerdings ergibt sich dabei eine recht aufwendige Hin- und Herrechnerei im Programm.

4. Programmtechnische Details

Die meisten Teile des Programms sollten in ihrem Ablauf relativ einfach zu verfolgen sein. Einige Stellen bedürfen allerdings eingehenderer Erklärung bzw. sind „kritisch":

Direkt-Modus: DOS prüft bei Befehlen und Eingaben, ob ein Applesoft-Programm läuft, und testet dazu die Zero-Page-Adressen $0033 (Prompt) und $0076 (CURLIN = momentane Applesoft-Programmzeilennummer). Falls diese Speicherstellen dieselben Werte wie im Direkt-Modus enthalten (Prompt = „Ü" und CURLIN = $FF), behandelt DOS jede Eingabe wie einen direkten Befehl und würde z.B. auf eine Dateinamen-Eingabe CATALOG mit einem CATALOG reagieren. Das

Setzen dieser beiden Speicherstellen ist ein eleganterer Weg als das „Abhängen" von DOS.

Volume-Directory: Nach der Meldung von RESTART wird das gesamte Volume-Directory (4 Blocks = $800 Bytes) von der ProDOS-Diskette eingelesen (Routine RDDIR) und danach über PRTDIR ausgegeben. Das Absuchen dieser vier Blocks nach (belegten) Einträgen (GET1ST und GETNXT) ist aus dem ProDOS-Urlader übernommen und ziemlich fehleranfällig. Jeder Eintrag hat eine Länge von $27 Bytes, wobei der erste Eintrag beim 4. Byte ($004) des jeweiligen Blocks beginnt. Daraus folgt: Der letzte Eintrag innerhalb eines Directory-Blocks beginnt mit dem Byte $1D8 und endet mit dem Byte $1FE – bei einer weiteren Erhöhung um $27 für den nächsten Eintrag verrechnet sich die Suchroutine, denn es kommt $1FF heraus. Hier steht in einem ProDOS-Directoryblock immer der Wert $00, und die Suchroutine „erkennt" daraus „gelöschter Eintrag" und nimmt eine erneute Erhöhung des Pointers vor mit dem Ergebnis $026 im nächsten Block, in dem die Einträge aber wieder mit dem 4. Byte beginnen. Aus diesem Grund muß jeweils am Ende eines Blocks, also nach $200 Bytes „von Hand" nachkorrigiert werden. Der Test auf „innerhalb Block" oder „nächster Block und Korrektur" ist dabei von der Adresse abhängig, ab der das Directory geladen wurde.

Index-Blocks: Über das Thema „Lesen einer ProDOS-Datei" im Zusammenhang mit Master-, Index- und Datenblocks ließen sich ganze Bücher schreiben. Hier sei nur angemerkt, daß ein Masterblock die Blocknummern von Indexblocks enthält, die ihrerseits wiederum die Blocknummern der einzelnen Datenblocks einer Datei enthalten. Je nach Dateigröße besteht eine ProDOS-Datei nur aus einem einzigen Datenblock (max. 512 Bytes), einem Indexblock mit max. 256 Datenblocks (max. 256 * 512 Byte) oder einem Masterblock, der max. 128 Indexblocknummern enthält. Die Blocknummer im Dateieintrag zeigt immer auf den „höchstwertigen" Block der Datei, d.h. bei kleinen Dateien direkt auf den einzigen Datenblock, ansonsten auf den Index- bzw. Masterblock.

Skewing: In der Routine RDBLOCK wird zuerst aus der Blocknummer die Track- und Sektornummer ermittelt. Dabei kommt jedoch bereits die physikalische Sektornummer heraus. Da die RWTS von DOS intern eine Umsetzung von logischer nach physikalischer Sektornummer vornimmt („Sektor Skewing"), muß diese Umsetzung vorher rückwärts vorgenommen werden, nämlich so, daß die RWTS nach der internen Umrechnung dann wieder den tatsächlich gewünschten Sektor liest.

Kurzhinweise

1. Zweck:
Konvertierung von ProDOS- in DOS-3.3-Dateien
2. Konfiguration:
Apple II+/e/c; DOS 3.3
3. Test:
BRUN PROTODOS
(von DOS-3.3-Diskette aus!)
4. Sammeldisk:
T.PROTODOS
(Big-Mac-Quelltext)
PROTODOS
(Maschinenprogramm)

PROTODOS

```
BSAVE PROTODOS, A$0803, L1584

                 1   ************************
                 2   *                      *
                 3   *    * PROTODOS *      *
                 4   *                      *
                 5   *  ProDOS -> DOS 3.3   *
                 6   *                      *
                 7   * Arne Schäpers 10/85 *
                 8   ************************
                 9   *
                10   MSGPNT   EQU  0          PTR für PRINT
                11   PTR      EQU  $05        diverse Zwecke
                12   IOBPNT   EQU  $07        PTR auf IOB-Liste
                13   TEMP     EQU  $09        Scratch
                14   *
                15   WNDLFT   EQU  $20        Textfenster links
                16   CH       EQU  $24        Curs. horizontal
                17   CV       EQU  $25        Curs. vertikal
                18   PREG     EQU  $48        Bug-Fix (RWTS)
                19   *
                20   IN       EQU  $200       Eingabe-Puffer
                21   *
                22   STYP     EQU  $300       ProDOS-Dateieintrag
                23   EOF      EQU  STYP+$15   (3 Byte)
                24   NAMBUF   EQU  $330       DOS-Filename
                25   TWORK    EQU  $360       FMan-Workspace
                26   *
                27   * PAGE-3-VEKTOREN:
                28   *
                29   DOSCOLD  EQU  $3D3       Kaltstart DOS
                30   FMAN     EQU  $3D6       FMan-Aufruf
                31   RWTS     EQU  $3D9       RWTS-Aufruf
                32   FMANLOC  EQU  $3DC       Adresse bestimmen
                33   RWTSLOC  EQU  $3E3       Adresse bestimmen
                34   *
                35   * DATEN/FILE-MANAGER-PUFFER:
                36   *
                37   TTSLIST  EQU  $0F00      TS-Puffer (FMan)
                38   TDBUF    EQU  $1000      Sec-Puffer (FMan)
                39   DIRBUF   EQU  $1100      ProDOS-Directory
                40   DTABUF   EQU  $1500      Datenpuffer-Start
                41   *
                42   KBD      EQU  $C000      Tastatur
                43   KBSTRB   EQU  $C010      Taste löschen
                44   *
                45   * MONITOR-AUFRUFE:
                46   *
                47   TEXT     EQU  $FB2F      24 * 40 Z
                48   HOME     EQU  $FC58
                49   RDKEY    EQU  $FD0C      1 Tast.-Zeichen
                50   GETLN    EQU  $FD6A      1 Tast.-Zeile
                51   CROUT    EQU  $FD8E      <CR>-Ausgabe
                52   PRBYTE   EQU  $FDDA      Byte ausgeben
                53   COUT     EQU  $FDED      Zeichenausgabe
                54   *
                55            ORG  $0803
                56   *
0803: A9 A0     57            LDA  #$A0
0805: 85 33     58            STA  $33        PROMPT und
0807: 85 76     59            STA  $76        CURLIN =>
0809: 20 2F FB  60            JSR  TEXT       "AP läuft"
080C: A9 95     61            LDA  #$95       CHR$ (21)
080E: 20 ED FD  62            JSR  COUT       => 80-Z aus
0811: 20 58 FC  63            JSR  HOME
0814: A9 0A     64            LDA  #10
0816: 85 25     65            STA  CV         VTAB 09
0818: A9 06     66            LDA  #$06
081A: 85 20     67            STA  WNDLFT     "HTAB 5"
081C: 20 24 0B  68            JSR  PRINT
081F: 8D        69            HEX  8D
0820: D0 D2 CF  70            ASC  "PRODOS -> DOS-KONVERTER"
0823: C4 CF D3 A0 AD BE A0 C4
082B: CF D3 AD CB CF CE D6 C5
0833: D2 D4 C5 D2
0837: 8D 8D     71            HEX  8D8D
0839: A0 A0 C1  72            ASC  "  ARNE SCHÄPERS 10/85"
083C: D2 CE C5 A0 D3 C3 C8 DB
0844: D0 C5 D2 D3 A0 B1 B0 AF
084C: B8 B5
084E: 8D 8D 8D  73            HEX  8D8D8D00
0851: 00
0852: A9 00     74            LDA  #$00       WNDLFT zurück
0854: 85 20     75            STA  WNDLFT
                76   *
```

```
Ø856: 2Ø 2B ØE  77            JSR  GETIOB      IOB-Adresse
Ø859: AØ ØC     78            LDY  #$ØC        RWTS-Parms
Ø85B: B1 Ø7     79   SETIOB   LDA  (IOBPNT),Y  kopieren
Ø85D: 99 78 ØD  8Ø            STA  IOB,Y       wg. Slotnummer
Ø86Ø: 88        81            DEY
Ø861: DØ F8     82            BNE  SETIOB
Ø863: A5 Ø8     83            LDA  IOBPNT+1    Bestimmung des
Ø865: 38        84            SEC              freien Platzes
Ø866: E9 21     85            SBC  #$21        -> norm. = $96(ØØ)
Ø868: E9 ØØ     86            SBC  #<DTABUF    Daten-Start
Ø86A: 29 FE     87            AND  #$FE        -> Geradzahl
Ø86C: 1Ø Ø2     88            BPL  SETFRE      wenn < $8ØØØ Byte
Ø86E: A9 8Ø     89            LDA  #$8Ø
                9Ø   *
Ø87Ø: 8D 32 ØA  91   SETFRE   STA  FRESPC      Datenpuffergröße
Ø873: 2Ø AE ØA  92            JSR  CRCONT      "RETURN -> Weiter"
                93   *
                94   ************************
                95   * RESTART NACH TRANSFER *
                96   ************************
                97   *
Ø876: 2Ø 58 FC  98   RESTART  JSR  HOME
Ø879: A9 ØA     99            LDA  #1Ø
Ø87B: 85 25     1ØØ           STA  CV          VTAB 11
Ø87D: 2Ø 24 ØB  1Ø1           JSR  PRINT
Ø88Ø: 8D        1Ø2           HEX  8D
Ø881: CC C5 C7  1Ø3           ASC  "LEGEN SIE..."
Ø884: C5 CE AØ D3 C9 C5 AE AE
Ø88C: AE
Ø88D: 8D 8D     1Ø4           HEX  8D8D
Ø88F: C4 C9 C5  1Ø5           ASC  "DIE PRODOS-DISKETTE
                                     IN LAUFWERK 2"
Ø892: AØ DØ D2 CF C4 CF D3 AD
Ø89A: C4 C9 D3 CB C5 D4 D4 C5
Ø8A2: AØ C9 CE AØ CC C1 D5 C6
Ø8AA: D7 C5 D2 CB AØ B2
Ø8BØ: 8D 8D     1Ø6           HEX  8D8D
Ø8B2: C5 C9 CE  1Ø7           ASC  "EINE DOS-DISKETTE
                                     IN LAUFWERK 1"
Ø8B5: C5 AØ C4 CF D3 AD C4 C9
Ø8BD: D3 CB C5 D4 D4 C5 AØ C9
Ø8C5: CE AØ CC C1 D5 C6 D7 C5
Ø8CD: D2 CB AØ B1
Ø8D1: 8D 8D     1Ø8           HEX  8D8D
Ø8D3: BC D2 C5  1Ø9           ASC  "<RETURN> = WEITER      <ESC>
                                     = ENDE"
Ø8D6: D4 D5 D2 CE BE AØ BD AØ
Ø8DE: D7 C5 C9 D4 C5 D2 AØ AØ
Ø8E6: AØ AØ AØ BC C5 D3 C3 BE
Ø8EE: AØ BD AØ C5 CE C4 C5
Ø8F5: 8D ØØ     11Ø           HEX  8DØØ
Ø8F7: AD ØØ CØ  111  GETCR1   LDA  KBD         das hier ist
Ø8FA: 1Ø FB     112           BPL  GETCR1      der einzige
Ø8FC: 8D 1Ø CØ  113           STA  KBSTRB      reguläre Ausgang
Ø8FF: C9 8D     114           CMP  #$8D
Ø9Ø1: FØ Ø7     115           BEQ  NEWDSK
Ø9Ø3: C9 9B     116           CMP  #$9B        <ESC>?
Ø9Ø5: DØ FØ     117           BNE  GETCR1
Ø9Ø7: 4C D3 Ø3  118           JMP  DOSCOLD
                119  *
Ø9ØA: 2Ø 58 FC  12Ø  NEWDSK   JSR  HOME
Ø9ØD: 2Ø E4 ØC  121           JSR  RDDIR       liest ProDOS-DIR
Ø91Ø: BØ Ø5     122           BCS  ERRVEC      => Lesefehler
Ø912: 2Ø 32 ØC  123           JSR  PRTDIR      druckt CATALOG
Ø915: 9Ø Ø3     124           BCC  GETPN       DIR nicht leer
Ø917: 4C 9D Ø9  125  ERRVEC   JMP  ERROR
                126  *
Ø91A: 2Ø 24 ØB  127  GETPN    JSR  PRINT
Ø91D: 8D        128           HEX  8D
Ø91E: DØ D2 CF  129           ASC  "PRODOS-DATEINAME:"
Ø921: C4 CF D3 AD C4 C1 D4 C5
Ø929: C9 CE C1 CD C5 BA
Ø92F: ØØ        13Ø           HEX  ØØ
Ø93Ø: 2Ø 6A FD  131           JSR  GETLN       Eingabe Dateiname
Ø933: 8A        132           TXA              Eingabe-Länge
Ø934: FØ 7Ø     133           BEQ  RSTVEC      => nur <CR>
Ø936: 85 Ø9     134           STA  TEMP        Namenslänge
Ø938: 2Ø 9F ØC  135           JSR  GET1ST      Vergleich mit
Ø93B: 2Ø C6 ØC  136  SCANDIR  JSR  CMPNAME     ProDOS-Einträgen
Ø93E: FØ 3Ø     137           BEQ  GOTPN
Ø94Ø: 2Ø A7 ØC  138           JSR  GETNXT      nächster Eintrag
Ø943: DØ F6     139           BNE  SCANDIR     Loop bis DIR-Ende
Ø945: 2Ø 24 ØB  14Ø           JSR  PRINT
Ø948: 8D 87     141           HEX  8D87        BELL
Ø94A: C6 C1 CC  142           ASC  "FALSCHER DATEITYP/
                                     NICHT GEFUNDEN!"
Ø94D: D3 C3 C8 C5 D2 AØ C4 C1
Ø955: D4 C5 C9 D4 D9 DØ AF CE
Ø95D: C9 C3 C8 D4 AØ C7 C5 C6
Ø965: D5 CE C4 C5 CE A1
Ø96B: 8D ØØ     143           HEX  8DØØ
Ø96D: 4C 1A Ø9  144           JMP  GETPN       => neue Eingabe
Ø97Ø: AC 23 11  145  GOTPN    LDY  DIRBUF+$23  Eintragslänge
Ø973: B1 Ø5     146  GOTP1    LDA  (PTR),Y     Kopie des
Ø975: 99 ØØ Ø3  147           STA  STYP,Y      gefundenen Eintrags
Ø978: 88        148           DEY              aus dem ProDOS-
Ø979: 1Ø F8     149           BPL  GOTP1       Directory
                15Ø  *
Ø97B: 2Ø 24 ØB  151  GETDN    JSR  PRINT
Ø97E: 8D        152           HEX  8D
Ø97F: C4 CF D3  153           ASC  "DOS-DATEINAME:"
Ø982: AD C4 C1 D4 C5 C9 CE C1
Ø98A: CD C5 BA
Ø98D: ØØ        154           HEX  ØØ
Ø98E: 2Ø 6A FD  155           JSR  GETLN
Ø991: 8A        156           TXA              Eingabe-Länge
Ø992: FØ 12     157           BEQ  RSTVEC      nur <CR>
Ø994: 2Ø 44 ØB  158           JSR  OPEN        OPEN DOS-Datei
Ø997: 9Ø 1Ø     159           BCC  GETOV       => existiert bereits
Ø999: C9 Ø6     16Ø           CMP  #$Ø6        FILE NOT FOUND?
Ø99B: FØ 5B     161           BEQ  MAKEFL      wenn ja: o.k.
                162  *
Ø99D: 2Ø 24 ØB  163  ERROR    JSR  PRINT
Ø9AØ: 8D 8D ØØ  164           HEX  8D8DØØ
Ø9A3: 2Ø AE ØA  165           JSR  CRCONT      "RETURN = Weiter"
Ø9A6: 4C 76 Ø8  166  RSTVEC   JMP  RESTART
                167  *
Ø9A9: 2Ø 96 ØB  168  GETOV    JSR  CLOSE       kein Disk I/O
Ø9AC: 2Ø 24 ØB  169           JSR  PRINT
Ø9AF: 8D        17Ø           HEX  8D
Ø9BØ: C4 C1 D4  171           ASC  "DATEI EXISTIERT. "
Ø9B3: C5 C9 AØ C5 D8 C9 D3 D4
Ø9BB: C9 C5 D2 D4 AE AØ
Ø9C1: DD C2 C5  172           ASC  "ÜBERSCHREIBEN (J/N): "
Ø9C4: D2 D3 C3 C8 D2 C5 C9 C2
Ø9CC: C5 CE AØ A8 CA AF CE A9
Ø9D4: BA AØ
Ø9D6: ØØ        173           HEX  ØØ
Ø9D7: 2Ø ØC FD  174  GOT01    JSR  RDKEY       "J" oder "N"
Ø9DA: 29 DF     175           AND  #$DF        => Großbuchstabe
Ø9DC: C9 CE     176           CMP  #"N"
Ø9DE: DØ Ø9     177           BNE  GOT02
Ø9EØ: 2Ø ED FD  178           JSR  COUT        "N" ausgeben
Ø9E3: 2Ø 8E FD  179           JSR  CROUT       + <CR>
Ø9E6: 4C 7B Ø9  18Ø           JMP  GETDN       -> neuer DOS-Name
Ø9E9: C9 CA     181  GOT02    CMP  #"J"
Ø9EB: DØ EA     182           BNE  GOT01
Ø9ED: 2Ø ED FD  183           JSR  COUT        "J" ausgeben
Ø9FØ: 2Ø 8E FD  184           JSR  CROUT       + <CR>
Ø9F3: 2Ø 9E ØB  185           JSR  DELETE      Datei löschen
Ø9F6: BØ A5     186           BCS  ERROR
                187  *
Ø9F8: AD 1Ø Ø3  188  MAKEFL   LDA  STYP+$1Ø    ProDOS-Filetyp
Ø9FB: 2Ø ØA ØB  189           JSR  GETTP2      => DOS-Filetyp
Ø9FE: B9 1A ØB  19Ø           LDA  DTYPES,Y
ØAØ1: 8D 43 ØB  191           STA  DOSTYP
ØAØ4: 2Ø 89 ØB  192           JSR  CREATE      erzeugt Datei
ØAØ7: BØ 94     193           BCS  ERROR
ØAØ9: AD 43 ØB  194           LDA  DOSTYP
ØAØC: FØ 25     195           BEQ  MOVE        -> TEXT
ØAØE: C9 Ø2     196           CMP  #$Ø2
ØA1Ø: FØ 12     197           BEQ  WRLEN       A: nur Länge
ØA12: AE 1F Ø3  198           LDX  STYP+$1F    BIN/SYS: Start-
ØA15: AC 2Ø Ø3  199           LDY  STYP+$2Ø    adresse
ØA18: AD 1Ø Ø3  2ØØ           LDA  STYP+$1Ø    ProDOS-Filetyp
ØA1B: 1Ø Ø4     2Ø1           BPL  WRSTART     -> ist BIN-Datei
ØA1D: A2 ØØ     2Ø2           LDX  #$ØØ        SYS-Datei: Start
ØA1F: AØ 2Ø     2Ø3           LDY  #$2Ø        auf $2ØØØ
ØA21: 2Ø C4 ØB  2Ø4  WRSTART  JSR  WR16        schreibt Startad.
ØA24: AE 15 Ø3  2Ø5  WRLEN    LDX  EOF
ØA27: AC 16 Ø3  2Ø6           LDY  EOF+1       Dateilänge
ØA2A: 2Ø C4 ØB  2Ø7           JSR  WR16        schreibt Länge
ØA2D: 9Ø Ø4     2Ø8           BCC  MOVE        "always"
                2Ø9  *
                21Ø  **************
                211  * MOVER LOOP *
                212  **************
                213  *
ØA2F: ØØ        214  DADDR    DFB  #<DTABUF    Datenstart
ØA3Ø: ØØ ØØ     215  RDCOUNT  DA   ØØØØ        gelesene Bytes
ØA32: ØØ        216  FRESPC   DFB  Ø           Datenpuffergröße
                217  *
                218  *
ØA33: 2Ø DA ØA  219  MOVE     JSR  SETRD       setzt DSTART+TSTEOF
ØA36: FØ 54     22Ø           BEQ  RDDONE      -> EOF ØØ ØØ ØØ
ØA38: 2Ø Ø4 ØD  221           JSR  RD1ST       liest 1.Block
ØA3B: BØ 68     222           BCS  RDDONZ      Disk-Fehler
ØA3D: EE 2F ØA  223  RWLOOP   INC  DADDR       Zieladresse + $2ØØ
ØA4Ø: EE 2F ØA  224           INC  DADDR
ØA43: EE 31 ØA  225           INC  RDCOUNT+1   Anzahl gelesene
ØA46: EE 31 ØA  226           INC  RDCOUNT+1   Bytes +$2ØØ
ØA49: 2Ø F1 ØA  227           JSR  TSTEOF      EOF überschritten?
```

```
ØA4C: 9Ø ØF      228            BCC  LIMIT      -> ja
ØA4E: AD 31 ØA   229            LDA  RDCOUNT+1  max. $8ØØØ Byte
ØA51: CD 32 ØA   23Ø            CMP  FRESPC     Puffergröße
ØA54: BØ 13      231            BCS  REWRITE    -> Puffer voll
                 232  *
ØA56: 2Ø 26 ØD   233  NXTRD     JSR  RDNEXT     nächster Block
ØA59: 9Ø E2      234            BCC  RWLOOP
ØA5B: BØ 48      235            BCS  RDDONZ
                 236  *
ØA5D: AD 15 Ø3   237  LIMIT     LDA  EOF        EOF überschritten.
ØA6Ø: 8D 3Ø ØA   238            STA  RDCOUNT    Begrenzung auf
ØA63: AD 16 Ø3   239            LDA  EOF+1      den EOF für WRITE
ØA66: 8D 31 ØA   24Ø            STA  RDCOUNT+1
                 241  *
ØA69: 2Ø F1 ØA   242  REWRITE   JSR  TSTEOF     EOF - RDCOUNT
ØA6C: 8E 15 Ø3   243            STX  EOF        = verbleibende
ØA6F: 8C 16 Ø3   244            STY  EOF+1      Anzahl von zu
ØA72: 8D 17 Ø3   245            STA  EOF+2      lesenden Bytes
ØA75: AD 3Ø ØA   246            LDA  RDCOUNT
ØA78: 38         247            SEC             FMan-Angabe ist
ØA79: E9 Ø1      248            SBC  #$Ø1       1 weniger als
ØA7B: AA         249            TAX             die tatsächliche
ØA7C: AD 31 ØA   25Ø            LDA  RDCOUNT+1  Anzahl der zu
ØA7F: E9 ØØ      251            SBC  #$ØØ       schreibenden Bytes!
ØA81: A8         252            TAY
ØA82: 2Ø DD ØB   253            JSR  WRITE      schreibt X-Y Byte
ØA85: BØ 1E      254            BCS  RDDONZ     -> FMan-Fehler
ØA87: 2Ø DA ØA   255            JSR  SETRD      RDCOUNT auf Ø
ØA8A: DØ CA      256            BNE  NXTRD      falls EOF noch > Ø
                 257  *
ØA8C: 2Ø 96 ØB   258  RDDONE    JSR  CLOSE      vor(!) CATALOG
ØA8F: BØ 17      259            BCS  RDQUIT     bei Fehler
ØA91: 2Ø 24 ØB   26Ø            JSR  PRINT
ØA94: 8D 84      261            HEX  8D84       <CR>, Control-D
ØA96: C3 C1 D4   262            ASC  "CATALOG,D1"
ØA99: C1 CC CF C7 AC C4 B1
ØAAØ: 8D ØØ      263            HEX  8DØØ
ØAA2: 4C A8 ØA   264            JMP  RDQUIT     kein 2. CLOSE
ØAA5: 2Ø 96 ØB   265  RDDONZ    JSR  CLOSE
ØAA8: 2Ø AE ØA   266  RDQUIT    JSR  CRCONT     "RETURN = Weiter"
ØAAB: 4C 76 Ø8   267            JMP  RESTART    das war's
                 268  *
                 269  ********************************
                 27Ø  *        Unterprogramme         *
                 271  ********************************
                 272  *
ØAAE: 2Ø 24 ØB   273  CRCONT    JSR  PRINT
ØAB1: 8D 8D ØØ   274            HEX  8D8DØØ
ØAB4: A9 Ø9      275            LDA  #Ø9        HTAB Ø8
ØAB6: 85 24      276            STA  CH
ØAB8: 2Ø 24 ØB   277            JSR  PRINT
ØABB: BC D2 C5   278            ASC  "<RETURN> = WEITER"
ØABE: D4 D5 D2 CE BE AØ BD AØ
ØAC6: D7 C5 C9 D4 C5 D2
ØACC: ØØ         279            HEX  ØØ
ØACD: AD ØØ CØ   28Ø  GETCR2    LDA  KBD        wartet auf
ØADØ: 1Ø FB      281            BPL  GETCR2     ein RETURN
ØAD2: 8D 1Ø CØ   282            STA  KBSTRB
ØAD5: C9 8D      283            CMP  #$8D
ØAD7: DØ F4      284            BNE  GETCR2
ØAD9: 6Ø         285            RTS
                 286  *
ØADA: A9 ØØ      287  SETRD     LDA  #ØØ        setzt RDCOUNT
ØADC: 8D 3Ø ØA   288            STA  RDCOUNT    zurück, Adresse
ØADF: 8D 31 ØA   289            STA  RDCOUNT+1  für den nächsten
ØAE2: A9 15      29Ø            LDA  #>DTABUF
ØAE4: 8D 2F ØA   291            STA  DADDR      Datenblock
ØAE7: AD 15 Ø3   292            LDA  EOF        und Test auf
ØAEA: ØD 16 Ø3   293            ORA  EOF+1      EOF = ØØ ØØ ØØ
ØAED: ØD 17 Ø3   294            ORA  EOF+2
ØAFØ: 6Ø         295            RTS
                 296  *
ØAF1: AD 15 Ø3   297  TSTEOF    LDA  EOF        Test, ob RD-
ØAF4: 38         298            SEC             COUNT > restliche
ØAF5: ED 3Ø ØA   299            SBC  RDCOUNT    Bytes im File
ØAF8: AA         3ØØ            TAX
ØAF9: AD 16 Ø3   3Ø1            LDA  EOF+1
ØAFC: ED 31 ØA   3Ø2            SBC  RDCOUNT+1  High-Byte
ØAFF: A8         3Ø3            TAY
ØBØØ: AD 17 Ø3   3Ø4            LDA  EOF+2
ØBØ3: E9 ØØ      3Ø5            SBC  #ØØ
ØBØ5: 6Ø         3Ø6            RTS
                 3Ø7  *
ØBØ6: AØ 1Ø      3Ø8  GETTYP    LDY  #$1Ø       Filetyp aus
ØBØ8: B1 Ø5      3Ø9            LDA  (PTR),Y    DIR-Eintrag.
ØBØA: AØ Ø4      31Ø  GETTP2    LDY  #$Ø4       Heraussuchen
ØBØC: D9 15 ØB   311  GTT1      CMP  PTYPES,Y   der unterstützten
ØBØF: FØ Ø3      312            BEQ  GOTTYP     ProDOS-Filetypen.
ØB11: 88         313            DEY             DOS-Filetyp
ØB12: DØ F8      314            BNE  GTT1       zuordnen
ØB14: 6Ø         315  GOTTYP    RTS
```


Jetzt purzeln die Preise!

ZUSATZ-KARTEN:

V-24-Schnittstelle 199,-
80-Zeichen-Karte m.Softswitch 236,-
Joy Stick De Luxe 59,-
68000 Intemex 1600,-
RGB Karte 239,-
Koppler dataphon m. FTZ 325,-
Centronics-Karte von Epson für Graphik .... 210,- für Text 145,-
Centronics-Schnittstelle für 2 Drucker gleichzeitig 129,-
Z-80-Karte 98,-
16 K-Language-Karte 98,-
Accelerator 3,6 MHz 950,-
PAL Karte 110,-
IEEE 488 312,-
Z 80 B Karte mit Software 919,-

Super-Eprommer
belegt keinen Slot, incl. Software für 2716-27128 ..... **239,-**

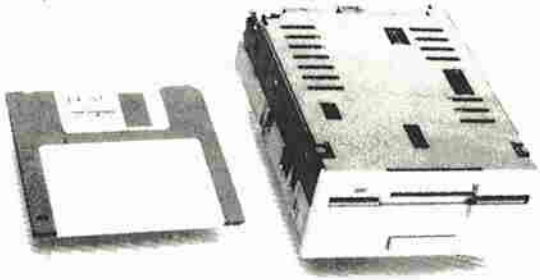
Floppy-Controller
FDC 4 für alle Laufwerke ..... 169,- Bausatz wie links 159,-
Leerplatine wie oben incl. Prom u. Eprom 98,-

Erphi-Controller **298,-**

Drucker-Spooler 64 kB,
fertig aufgebaut incl. Netzteil u. Gehäuse **340,-**

Preh Commander Keyboards
Wir bieten Ihnen die **Preh-Qualität** auch für Apple. AK 88 spez. mit Gehäuse, Anschlußkabel, Zehner-Tastenfeld, dt. Zeichensatz, Sondertasten für Ctrl-Codes und Rechenfunktionen **339,-**
Preh Commander Keyboard, frei programmierbar bis zu 10 Ebenen, pro Taste bis zu 250 Zeichen ..... nur **599,-**

TEAC 3½" Laufwerk FD 35 F **498,-**
Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur .....



TEAC FD 55 AV 1 x 40 Track .... 395,- TEAC FD 55 BV 2 x 40 Track 460,-
TEAC FD 55 EV 1 x 80 Track .... 445,- TEAC FD 55 FV 2 x 80 Track 449,-

Apple®-kompatibles Laufwerk incl. Gehäuse + Kabel 599,-
320 KB Laufwerk für IIc **948,-**
640 KB Laufwerk für IIc **1088,-**

EPSON DRUCKER

EPSON FX 80 ..... 1670,- EPSON FX 100 2159,-
EPSON RX 80 ..... 1079,- EPSON RX 80 FT 1295,-

Die Microfloppy mit Zukunft:
Speicherkapazität: 2 x 1 MByte formatiert: 2 x 640 kByte. Anschlußfertig mit PROM-residenter Patchsoftware für CP/M 2.2, Apple DOS 3.3, DiversiDOS 2-C, 4-C (DD MOVER), Apple Pascal 1.1, Pascal 1.2, Pro-DOS 1.0.1, 1.1, 1.1.1 zum Preis von **1598,-**
Low Power Version 1698,-

10 MB Winchester
mit Software für DOS 3.3, CP/M 2.20, Pascal, Pro-DOS, incl. Controler und Gehäuse **3990,-**

Sonderangebot
Chinon Laufwerk für II + IIe, incl. Kabel u. Gehäuse ..... jetzt nur **339,-**

Gesamt-Preisliste anfordern! Preise inclusive gesetzlicher Mehrwertsteuer.
Händlerpreisliste bitte schriftlich anfordern!

UEDING electronics
Holtewiese 2
5750 Menden 1
DFÜ 02373/66877
Tel. 02373/63159

```
                  316  *
ØB15: ØØ Ø4 FC    317  PTYPES   DFB  Ø,4,$FC,6,$FF TXT/BAS/BIN/SYS
ØB18: Ø6 FF
ØB1A: FF ØØ Ø2    318  DTYPES   DFB  $FF,Ø,2,4,4 X/T/A/B/B
ØB1D: Ø4 Ø4
ØB1F: AØ D4 C1    319  DCHARS   ASC  " TABB"
ØB22: C2 C2
                  32Ø  *
                  321  * PRINT ROUTINE nach Andy Hertzfeld...
                  322  *
ØB24: 68          323  PRINT    PLA                Returnadresse
ØB25: 85 ØØ       324          STA  MSGPNT        zeigt auf den
ØB27: 68          325          PLA                auszugebenden
ØB28: 85 Ø1       326          STA  MSGPNT+1      Text selber
ØB2A: AØ Ø1       327          LDY  #$Ø1
ØB2C: B1 ØØ       328  PRT1     LDA  (MSGPNT),Y
ØB2E: FØ Ø6       329          BEQ  PRTZ          <ØØ> = Textende
ØB3Ø: 2Ø ED FD    33Ø          JSR  COUT
ØB33: C8          331          INY
ØB34: DØ F6       332          BNE  PRT1
ØB36: 18          333  PRTZ     CLC
ØB37: 98          334          TYA
ØB38: 65 ØØ       335          ADC  MSGPNT        der ausgegebene
ØB3A: A8          336          TAY                Text wird über-
ØB3B: A5 Ø1       337          LDA  MSGPNT+1      sprungen (Erhö-
ØB3D: 69 ØØ       338          ADC  #$ØØ          hung der Return-
ØB3F: 48          339          PHA                adresse)
ØB4Ø: 98          34Ø          TYA
ØB41: 48          341          PHA
ØB42: 6Ø          342          RTS
                  343  *
                  344  ************************
                  345  * FILE MANAGER-AUFRUFE *
                  346  ************************
                  347  *
ØB43: ØØ          348  DOSTYP   DFB  Ø            DOS-Dateityp
                  349  *
ØB44: AØ ØØ       35Ø  OPEN     LDY  #$ØØ
ØB46: 8C 23 ØC    351          STY  TVOLNO        Volume-Nummer
ØB49: 8C 22 ØC    352          STY  TRECLEN+1     Recordlänge High
ØB4C: C8          353          INY
ØB4D: 8C 21 ØC    354          STY  TRECLEN       => $ØØØ1
ØB5Ø: 8C 24 ØC    355          STY  TDRIVE        Drive 1
ØB53: AD 79 ØD    356          LDA  IOB+1         RWTS-Slotnummer
ØB56: 4A          357          LSR  A
ØB57: 4A          358          LSR  A
ØB58: 4A          359          LSR  A
ØB59: 4A          36Ø          LSR  A             / 16
ØB5A: 8D 25 ØC    361          STA  TSLOT
ØB5D: A9 Ø3       362          LDA  #>NAMBUF      Adresse des
ØB5F: AØ 3Ø       363          LDY  #NAMBUF       Dateinamens
ØB61: 8D 28 ØC    364          STA  TNAME+1
ØB64: 8C 27 ØC    365          STY  TNAME
                  366  *
ØB67: AØ 1E       367          LDY  #3Ø
ØB69: A9 AØ       368          LDA  #$AØ
ØB6B: 99 3Ø Ø3    369  CLRNAME  STA  NAMBUF,Y      Der Dateinamen-
ØB6E: 88          37Ø          DEY                bereich wird
ØB6F: 1Ø FA       371          BPL  CLRNAME       auf " " gelöscht
ØB71: CA          372          DEX
ØB72: BD ØØ Ø2    373  MVNAME   LDA  IN,X          und der einge-
ØB75: 9D 3Ø Ø3    374          STA  NAMBUF,X      gebene Name wird
ØB78: CA          375          DEX                hineingeschrieben
ØB79: 1Ø F7       376          BPL  MVNAME
                  377  *
ØB7B: A9 Ø1       378          LDA  #$Ø1          Kommando: OPEN
ØB7D: AA          379          TAX                => kein CREATE
ØB7E: 2Ø FA ØB    38Ø          JSR  CALLFMAN
ØB81: 9Ø Ø5       381          BCC  FOPEN         => Datei existiert
ØB83: C9 Ø6       382          CMP  #$Ø6
ØB85: DØ 1F       383          BNE  FMERR         -> nicht NOT FOUND
ØB87: 38          384          SEC
ØB88: 6Ø          385  FOPEN    RTS
                  386  *
                  387  *
ØB89: 8D 26 ØC    388  CREATE   STA  TTYPE         DOS-Filetyp
ØB8C: A9 Ø1       389          LDA  #$Ø1          Kommando: OPEN
ØB8E: A2 ØØ       39Ø          LDX  #$ØØ          CREATE erlaubt
ØB9Ø: 2Ø FA ØB    391          JSR  CALLFMAN
ØB93: BØ 11       392          BCS  FMERR
ØB95: 6Ø          393          RTS
                  394  *
                  395  *
ØB96: A9 Ø2       396  CLOSE    LDA  #Ø2           Kommando: CLOSE
ØB98: 2Ø FA ØB    397          JSR  CALLFMAN
ØB9B: BØ Ø9       398          BCS  FMERR
ØB9D: 6Ø          399          RTS
ØB9E: A9 Ø5       4ØØ  DELETE   LDA  #Ø5           Kommando: DELETE
ØBAØ: 2Ø FA ØB    4Ø1          JSR  CALLFMAN
ØBA3: BØ Ø1       4Ø2          BCS  FMERR
ØBA5: 6Ø          4Ø3          RTS
                  4Ø4  *
                  4Ø5  *
ØBA6: 48          4Ø6  FMERR    PHA                Fehlernummer
ØBA7: 2Ø 24 ØB    4Ø7          JSR  PRINT
ØBAA: 8D 87       4Ø8          HEX  8D87
ØBAC: C6 CD C1    4Ø9          ASC  "FMAN-FEHLER: $"
ØBAF: CE AD C6 C5 C8 CC C5 D2
ØBB7: BA AØ A4
ØBBA: ØØ          41Ø          HEX  ØØ
ØBBB: 68          411          PLA
ØBBC: 2Ø DA FD    412          JSR  PRBYTE        druckt Hexzahl
ØBBF: 2Ø 8E FD    413          JSR  CROUT         <CR> danach
ØBC2: 38          414          SEC
ØBC3: 6Ø          415          RTS
                  416  *
                  417  *
ØBC4: 84 Ø9       418  WR16     STY  TEMP
ØBC6: 8E 27 ØC    419          STX  TNAME         WR16 schreibt
ØBC9: A9 Ø1       42Ø          LDA  #Ø1           zwei Byte, dabei
ØBCB: 8D 2Ø ØC    421          STA  SUBCODE       findet kein
ØBCE: A9 Ø4       422          LDA  #$Ø4          Disk-I/O statt
ØBDØ: 2Ø FA ØB    423          JSR  CALLFMAN      -> 1. Byte
ØBD3: A5 Ø9       424          LDA  TEMP
ØBD5: 8D 27 ØC    425          STA  TNAME         2. Byte
ØBD8: A9 Ø4       426          LDA  #$Ø4
ØBDA: 4C FA ØB    427          JMP  CALLFMAN
                  428  *
                  429  *
ØBDD: 8E 25 ØC    43Ø  WRITE    STX  TWRNUM        schreibt X-Y Byte
ØBEØ: 8C 26 ØC    431          STY  TWRNUM+1      in die DOS-Datei
ØBE3: A9 15       432          LDA  #>DTABUF
ØBE5: AØ ØØ       433          LDY  #Ø
ØBE7: 8D 28 ØC    434          STA  TWRADDR+1     Daten werden ab
ØBEA: 8C 27 ØC    435          STY  TWRADDR       DTABUF geschrieben
ØBED: A9 Ø2       436          LDA  #$Ø2
ØBEF: 8D 2Ø ØC    437          STA  SUBCODE       => WRITE RANGE
ØBF2: A9 Ø4       438          LDA  #$Ø4
ØBF4: 2Ø FA ØB    439          JSR  CALLFMAN      schreibt RANGE
ØBF7: BØ AD       44Ø          BCS  FMERR
ØBF9: 6Ø          441          RTS
                  442  *
                  443  *
ØBFA: 8D 1F ØC    444  CALLFMAN STA  OPCODE        Aufruf mit OPC
ØBFD: 2Ø DC Ø3    445          JSR  FMANLOC
ØCØØ: 85 Ø6       446          STA  PTR+1         -> zur Parmliste
ØCØ2: 84 Ø5       447          STY  PTR
ØCØ4: AØ 11       448          LDY  #$11
ØCØ6: B9 1F ØC    449  SETFMAN  LDA  OPCODE,Y      Parmliste wird
ØCØ9: 91 Ø5       45Ø          STA  (PTR),Y       in FMan kopiert
ØCØB: 88          451          DEY
ØCØC: 1Ø F8       452          BPL  SETFMAN
ØCØE: 2Ø D6 Ø3    453          JSR  FMAN          FMan-Aufruf
ØC11: AØ 11       454          LDY  #$11
ØC13: B1 Ø5       455  GETFMAN  LDA  (PTR),Y       Rückkopie der
ØC15: 99 1F ØC    456          STA  OPCODE,Y      FMan-Parmliste
ØC18: 88          457          DEY
ØC19: 1Ø F8       458          BPL  GETFMAN
ØC1B: AD 29 ØC    459          LDA  RTNCODE       Carry unverändert!
ØC1E: 6Ø          46Ø          RTS
                  461  *
                  462  OPCODE   DS   1
                  463  SUBCODE  DS   1
ØC21: Ø1 ØØ       464  TRECLEN  DA   $ØØØ1         Recordlänge
ØC23: ØØ          465  TVOLNO   DFB  Ø              Volume-Nummer
ØC24: Ø1          466  TDRIVE   DFB  Ø1             Drive 1
ØC25: Ø6          467  TSLOT    DFB  Ø6             aus RWTS gesetzt!
                  468  TTYPE    DS   1              DOS-Filetyp
ØC27: 3Ø Ø3       469  TNAME    DA   NAMBUF         Startadresse Name
                  47Ø  RTNCODE  DS   2
ØC2B: 6Ø Ø3       471          DA   TWORK          Workspace
ØC2D: ØØ ØF       472          DA   TTSLIST        $1ØØ Bytes
ØC2F: ØØ 1Ø       473          DA   TDBUF          $1ØØ Bytes
                  474  *
                  475  TWRNUM   EQU  TSLOT          für WRITE RANGE
                  476  TWRADDR  EQU  TNAME          für WRITE RANGE
                  477  *
                  478  ************************************
                  479  * ProDOS-Directory lesen & absuchen *
                  48Ø  ************************************
                  481  *
                  482  CRFLAG   DS   1              <CR> oder CHR$ (128)
                  483  *
ØC32: A9 8D       484  PRTDIR   LDA  #$8D
ØC34: 8D 31 ØC    485          STA  CRFLAG
ØC37: 2Ø 9F ØC    486          JSR  GET1ST         PTR -> 1. Eintrag
ØC3A: DØ 18       487          BNE  PRTENT
ØC3C: 2Ø 24 ØB    488          JSR  PRINT
ØC3F: 8D 87       489          HEX  8D87
ØC41: C4 C9 D2    49Ø          ASC  "DIRECTORY LEER!"
ØC44: C5 C3 D4 CF D2 D9 AØ CC
ØC4C: C5 C5 D2 A1
ØC5Ø: 8D ØØ       491          HEX  8DØØ
```

```
0C52: 38          492            SEC                 => kein einziger
0C53: 60          493            RTS                 Eintrag vorhanden
                  494  *
0C54: 29 0F       495  PRTENT    AND  #$0F           => Namenslänge
0C56: AA          496            TAX
0C57: 20 06 0B    497            JSR  GETTYP         ProDOS-> DOS-Filetyp
0C5A: B9 1F 0B    498            LDA  DCHARS,Y       " ", T, A, B
0C5D: 20 ED FD    499            JSR  COUT
0C60: A9 A0       500            LDA  #$A0           Space danach
0C62: 20 ED FD    501            JSR  COUT
0C65: A0 01       502            LDY  #$01
0C67: B1 05       503  PRTNAM    LDA  (PTR),Y        Ausgabe des
0C69: 09 80       504            ORA  #$80           Dateinamens
0C6B: 20 ED FD    505            JSR  COUT
0C6E: C8          506            INY
0C6F: CA          507            DEX                 Zähler für Länge
0C70: D0 F5       508            BNE  PRTNAM
0C72: A9 14       509            LDA  #20            zwei Einträge
0C74: 85 24       510            STA  CH             pro Zeile, CR
0C76: 20 83 0C    511            JSR  DOCR           nach jedem 2.Eintrag
0C79: 20 A7 0C    512            JSR  GETNXT         nächster Eintrag
0C7C: D0 D6       513            BNE  PRTENT         bis DIR-Ende
0C7E: 20 83 0C    514            JSR  DOCR
0C81: 18          515            CLC
0C82: 60          516            RTS
                  517  *
0C83: AD 31 0C    518  DOCR      LDA  CRFLAG         "Toggle": einmal
0C86: 49 0D       519            EOR  #$0D           wird CHR$(128),
0C88: 8D 31 0C    520            STA  CRFLAG         danach wieder <CR>
0C8B: 20 ED FD    521            JSR  COUT           ausgegeben
0C8E: C9 8D       522            CMP  #$8D           war <CR>?
0C90: D0 0C       523            BNE  CRDONE
0C92: A5 25       524            LDA  CV             über Zeile
0C94: C9 13       525            CMP  #19            20 hinaus?
0C96: 90 06       526            BCC  CRDONE
0C98: 20 AE 0A    527            JSR  CRCONT         "RETURN = Weiter"
0C9B: 20 58 FC    528            JSR  HOME
0C9E: 60          529  CRDONE    RTS
                  530  *
                  531  *
0C9F: A9 11       532  GET1ST    LDA  #>DIRBUF       holt den 1.
0CA1: 85 06       533            STA  PTR+1          besetzten Eintrag
0CA3: A9 04       534            LDA  #$04           nach Vol-Name
0CA5: D0 02       535            BNE  GTN1           "always"
                  536  *
0CA7: A5 05       537  GETNXT    LDA  PTR            holt folgende
0CA9: 18          538  GTN1      CLC                 Einträge aus DIR
0CAA: 6D 23 11    539            ADC  DIRBUF+$23     Eintragslänge
0CAD: A8          540            TAY
0CAE: 90 0D       541            BCC  SETPTR
0CB0: E6 06       542            INC  PTR+1
0CB2: A5 06       543            LDA  PTR+1          neuer Block?
0CB4: 4A          544            LSR  A
0CB5: 90 06       545            BCC  SETPTR         -> innerhalb Block
0CB7: C9 0C       546            CMP  #$0C           DIR-Ende/2
0CB9: F0 0A       547            BEQ  GOTENT         -> DIR-Ende
0CBB: A0 04       548            LDY  #$04           nächster DIR-Block
0CBD: 84 05       549  SETPTR    STY  PTR
0CBF: A0 00       550            LDY  #$00           Test 1.Byte
0CC1: B1 05       551            LDA  (PTR),Y        des Eintrags:
0CC3: F0 E2       552            BEQ  GETNXT         -> 00: gelöscht
0CC5: 60          553  GOTENT    RTS                 "NE": Eintrag
                  554  *
                  555  *
0CC6: 29 0F       556  CMPNAME   AND  #$0F           Aufruf mit 1.Byte
0CC8: C5 09       557            CMP  TEMP           des Eintrags
0CCA: D0 17       558            BNE  NOTSAME        Längen ungleich
0CCC: A8          559            TAY
0CCD: AA          560            TAX
0CCE: CA          561  CMPLOOP   DEX                 Vergleich DIR-
0CCF: B1 05       562            LDA  (PTR),Y        Eintrag mit
0CD1: 09 80       563            ORA  #$80           eingegebenem
0CD3: DD 00 02    564            CMP  IN,X           Dateinamen und
0CD6: D0 0B       565            BNE  NOTSAME
0CD8: 88          566            DEY                 Test auf gültigen
0CD9: D0 F3       567            BNE  CMPLOOP        Dateityp für
0CDB: 20 06 0B    568            JSR  GETTYP         Konversion
0CDE: B9 1A 0B    569            LDA  DTYPES,Y       < $0F, wenn o.k.
0CE1: 29 F0       570            AND  #$F0           -> EQ, wenn o.k.
0CE3: 60          571  NOTSAME   RTS                 "EQ": gefunden
                  572  *
                  573  *
                  574  *
0CE4: A9 04       575  RDDIR     LDA  #$04           4 Blocks
0CE6: 85 09       576            STA  TEMP           Herunterzähler
0CE8: A9 11       577            LDA  #>DIRBUF       Zieladresse High
0CEA: A2 02       578            LDX  #$02
0CEC: A0 00       579            LDY  #$00           Block $0002
0CEE: 20 85 0D    580  RDD1      JSR  RDBLOCK
0CF1: B0 0E       581            BCS  DIREND         -> RWTS-Fehler
0CF3: AD 81 0D    582            LDA  RDADDR+1       wurde erhöht
```


prodos

MERLIN

von Glen Bredon

Der professionelle Macro-Assembler für die Apple II-Familie!
Neben allen Standard-Features bietet MERLIN u.a.:

- komfortabler Editor mit globalen Such- und Ersetzfunktionen
- liest und schreibt DOS 3.3 Text- und Binärfiles
- unterstützt 6502- und 65C02-Opcodes
- beinhaltet eine Bibliothek mit fertigen Unterprogrammen
- enthält einen Disassembler mit eigener Label-Bibliothek
- kompatibel mit vielen 80-Zeichenkarten und natürlich mit Apple //e und Apple //c

Jetzt mit deutschem Handbuch!

ISBN 3-89058-024-6 DM 198,--

Bei Ihrem Apple-Händler, in guten Buchhandlungen, oder direkt vom Verlag:

Ampersand*, Teltower Damm 168,
D-1000 Berlin 37, (030) 815 80 69

**vormals Pandabooks*

Bestellcoupon

Name .....
Anschrift .....
.....

○ *V-Scheck liegt bei (spesenfreie Lieferung)*
○ *per Nachnahme (zzgl. DM 3,– Versandspesen)*

____ Ex. MERLIN à DM 198,--

```
ØCF6: AE 76 ØD  583            LDX  BLOCKNO
ØCF9: AC 77 ØD  584            LDY  BLOCKNO+1
ØCFC: E8        585            INX                 nächster Block
ØCFD: C6 Ø9     586            DEC  TEMP
ØCFF: DØ ED     587            BNE  RDD1
ØDØ1: 6Ø        588  DIREND    RTS
                589  *
                59Ø  *
                591  ****************************
                592  * PRODOS-DATEI EINLESEN    *
                593  ****************************
                594  *
ØDØ2: ØØ        595  MINDEX    DFB  Ø              Index im Master
ØDØ3: ØØ        596  IINDEX    DFB  Ø              Index im Index
                597  *
ØDØ4: A9 ØØ     598  RD1ST     LDA  #Ø             liest 1.Block
ØDØ6: 8D Ø2 ØD  599            STA  MINDEX         Master-Index
ØDØ9: 8D Ø3 ØD  6ØØ            STA  IINDEX         Index-Index
ØDØC: AE 11 Ø3  6Ø1            LDX  STYP+$11
ØDØF: AC 12 Ø3  6Ø2            LDY  STYP+$12       Nummer 1.Block
ØD12: AD ØØ Ø3  6Ø3            LDA  STYP
ØD15: 29 FØ     6Ø4            AND  #$FØ           => Storage Type
ØD17: C9 1Ø     6Ø5            CMP  #$1Ø
ØD19: FØ 3Ø     6Ø6            BEQ  TYPE1          nur 1 Datenblock
ØD1B: C9 2Ø     6Ø7            CMP  #$2Ø
ØD1D: FØ 16     6Ø8            BEQ  TYPE2          1 Indexblock
ØD1F: A9 11     6Ø9            LDA  #>DIRBUF
ØD21: 2Ø 85 ØD  61Ø            JSR  RDBLOCK        Master-Block
ØD24: BØ 4F     611            BCS  RDEND          in DIR-Bereich
                612  *
ØD26: AE Ø3 ØD  613  RDNEXT    LDX  IINDEX         Index im Index-
ØD29: DØ 19     614            BNE  RDDATA         Block: wenn Ø=>
ØD2B: AE Ø2 ØD  615            LDX  MINDEX         dann nächster
ØD2E: BC ØØ 12  616            LDY  DIRBUF+$1ØØ,X  Indexblock
ØD31: BD ØØ 11  617            LDA  DIRBUF,X       -> Blocknummer
ØD34: AA        618            TAX
ØD35: A9 13     619  TYPE2     LDA  #>DIRBUF+$2ØØ
ØD37: 2Ø 85 ØD  62Ø            JSR  RDBLOCK        Indexblock
ØD3A: BØ 39     621            BCS  RDEND
ØD3C: EE Ø2 ØD  622            INC  MINDEX         Index im Master
ØD3F: A2 ØØ     623            LDX  #$ØØ
ØD41: 8E Ø3 ØD  624            STX  IINDEX         Index im Index
ØD44: BC ØØ 14  625  RDDATA    LDY  DIRBUF+$3ØØ,X
ØD47: BD ØØ 13  626            LDA  DIRBUF+$2ØØ,X
ØD4A: AA        627            TAX                 Blocknummer
ØD4B: AD 2F ØA  628  TYPE1     LDA  DADDR          Zieladresse
ØD4E: 2Ø 85 ØD  629            JSR  RDBLOCK
ØD51: BØ 22     63Ø            BCS  RDEND
ØD53: EE Ø3 ØD  631            INC  IINDEX         Index-Index+1
ØD56: AD 76 ØD  632            LDA  BLOCKNO        war das ein
ØD59: ØD 77 ØD  633            ORA  BLOCKNO+1      leerer Block?
ØD5C: FØ 17     634            BEQ  RDEND          -> kein SETMSB
ØD5E: AC 43 ØB  635            LDY  DOSTYP         TXT-Datei?
ØD61: DØ 12     636            BNE  RDEND
ØD63: 84 Ø5     637            STY  PTR            PTR+1 ist gesetzt
ØD65: A2 Ø2     638            LDX  #$Ø2
ØD67: B1 Ø5     639  SETMSB    LDA  (PTR),Y        TXT: MSB wird
ØD69: Ø9 8Ø     64Ø            ORA  #$8Ø           innerhalb des
ØD6B: 91 Ø5     641            STA  (PTR),Y        Blocks gesetzt
ØD6D: C8        642            INY
ØD6E: DØ F7     643            BNE  SETMSB
ØD7Ø: E6 Ø6     644            INC  PTR+1
ØD72: CA        645            DEX                 Zähler: $2ØØ Byte
ØD73: DØ F2     646            BNE  SETMSB
ØD75: 6Ø        647  RDEND     RTS
                648  *
                649  *
                65Ø  ****************************
                651  * ProDOS-Blockread via RWTS *
                652  ****************************
                653  *
ØD76: ØØ ØØ     654  BLOCKNO   DA   $ØØØØ          ProDOS-Block
                655  *
ØD78: Ø1 6Ø     656  IOB       HEX  Ø16Ø           Type + Slot*16
ØD7A: Ø1        657  DRIVE     DFB  Ø1
ØD7B: ØØ        658  VOLNO     DFB  ØØ             Volume-Nummer
ØD7C: ØØ        659  TRACK     DFB  ØØ
ØD7D: ØØ        66Ø  SECTOR    DFB  ØØ
                661            DS   2              DCT-Pointer
ØD8Ø: ØØ 11     662  RDADDR    DA   DIRBUF         READ-Zieladresse
                663            DS   2              Sector Count
ØD84: Ø1        664  IOBCMD    DFB  Ø1             Kommando: READ
                665  *
                666  ****************************
                667  *
ØD85: 8E 76 ØD  668  RDBLOCK   STX  BLOCKNO        Aufruf mit Blockno
ØD88: 8C 77 ØD  669            STY  BLOCKNO+1      in X(Low), Y(High)
ØD8B: 8D 81 ØD  67Ø            STA  RDADDR+1
ØD8E: 85 Ø6     671            STA  PTR+1          f. SETMSB, CLRBLK
ØD9Ø: 98        672            TYA
ØD91: ØD 76 ØD  673            ORA  BLOCKNO        Blockno = ØØ ØØ?
```

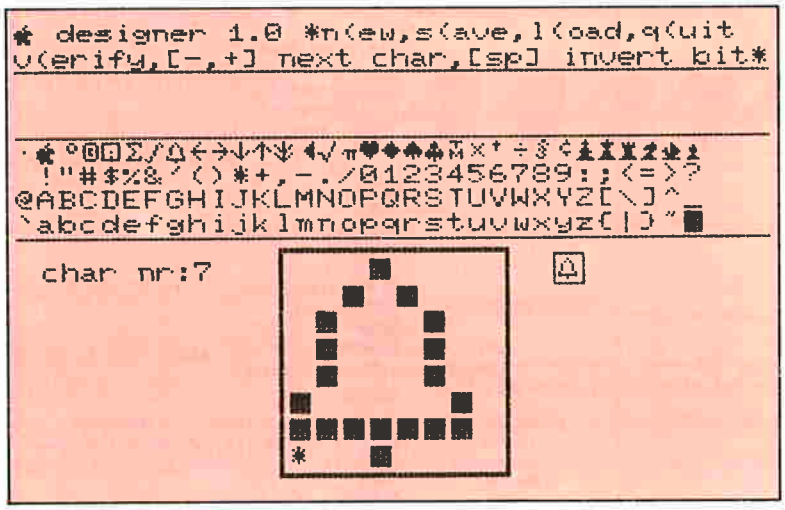
```
ØD94: DØ 11     674            BNE  XLATE
ØD96: A8        675            TAY                 = ØØ
ØD97: 84 Ø5     676            STY  PTR
ØD99: A2 Ø2     677            LDX  #$Ø2           $2ØØ Bytes löschen
ØD9B: 91 Ø5     678  CLRBLK    STA  (PTR),Y
ØD9D: C8        679            INY                 nicht belegte
ØD9E: DØ FB     68Ø            BNE  CLRBLK         Blocks ("sparse")
ØDAØ: E6 Ø6     681            INC  PTR+1          werden gelöscht
ØDA2: CA        682            DEX
ØDA3: DØ F6     683            BNE  CLRBLK
ØDA5: 18        684            CLC
ØDA6: 6Ø        685            RTS
                686  *
ØDA7: 8C 7C ØD  687  XLATE     STY  TRACK          Blockno High
ØDAA: 8A        688            TXA                 Blockno Low
ØDAB: AØ Ø5     689            LDY  #$Ø5
ØDAD: ØA        69Ø  SETTRK    ASL  A              5 Shifts links
ØDAE: 2E 7C ØD  691            ROL  TRACK          => Teilung / 8
ØDB1: 88        692            DEY
ØDB2: DØ F9     693            BNE  SETTRK
ØDB4: 8A        694            TXA                 Blockno Low
ØDB5: 29 Ø7     695            AND  #$Ø7
ØDB7: AA        696            TAX                 -> phys. Sektorno
ØDB8: BD ØB ØE  697            LDA  RETRANS,X      für RWTS: Zurück-
ØDBB: 8D 7D ØD  698            STA  SECTOR         übersetzen!
                699  *
ØDBE: AØ ØØ     7ØØ            LDY  #$ØØ
ØDCØ: 8C 7B ØD  7Ø1            STY  VOLNO          Volume-Nummer ØØ
ØDC3: 8C 8Ø ØD  7Ø2            STY  RDADDR         Zieladresse Low
ØDC6: C8        7Ø3            INY
ØDC7: 8C 84 ØD  7Ø4            STY  IOBCMD         Kommando: READ
ØDCA: C8        7Ø5            INY
ØDCB: 8C 7A ØD  7Ø6            STY  DRIVE          Drive 2
ØDCE: 2Ø 13 ØE  7Ø7            JSR  RDSEC          liest 1. Sektor
ØDD1: BØ 18     7Ø8            BCS  RDERR          -> RWTS-Fehler
ØDD3: EE 81 ØD  7Ø9            INC  RDADDR+1       Zieladresse+$1ØØ
ØDD6: AD 7D ØD  71Ø            LDA  SECTOR
ØDD9: E9 Ø1     711            SBC  #Ø1            Ø D B 9 7 5 3 1
ØDDB: 69 ØØ     712            ADC  #ØØ            wird zu:
ØDDD: 29 ØF     713            AND  #$ØF           E C A 8 6 4 2 F
ØDDF: 8D 7D ØD  714            STA  SECTOR         -> 2. Sektor
ØDE2: 2Ø 13 ØE  715            JSR  RDSEC          liest 2.Sektor
ØDE5: BØ Ø4     716            BCS  RDERR
ØDE7: EE 81 ØD  717            INC  RDADDR+1
ØDEA: 6Ø        718            RTS
                719  *
ØDEB: 2Ø 24 ØB  72Ø  RDERR     JSR  PRINT
ØDEE: 8D 87     721            HEX  8D87
ØDFØ: D2 D7 D4  722            ASC  "RWTS-FEHLER: $"
ØDF3: D3 AD C6 C5 C8 CC C5 D2
ØDFB: BA AØ A4
ØDFE: ØØ        723            HEX  ØØ
ØDFF: AØ ØD     724            LDY  #13
ØE Ø1: B1 Ø7    725            LDA  (IOBPNT),Y
ØE Ø3: 2Ø DA FD 726            JSR  PRBYTE         Fehlernummer
ØE Ø6: 2Ø 8E FD 727            JSR  CROUT          <CR> danach
ØE Ø9: 38       728            SEC
ØE ØA: 6Ø       729            RTS
                73Ø  *
ØE ØB: ØØ ØD ØB 731  RETRANS   DFB  Ø,$D,$B,9,7,5,3,1 rückwärts
ØE ØE: Ø9 Ø7 Ø5 Ø3 Ø1
                732  *
ØE13: 2Ø 2B ØE  733  RDSEC     JSR  GETIOB         setzt IOBPNT
ØE16: AØ ØC     734            LDY  #$ØC
ØE18: B9 78 ØD  735  MVIOB     LDA  IOB,Y
ØE1B: 91 Ø7     736            STA  (IOBPNT),Y
ØE1D: 88        737            DEY
ØE1E: DØ F8     738            BNE  MVIOB
ØE2Ø: 2Ø E3 Ø3  739            JSR  RWTSLOC        lädt Y-A
ØE23: 2Ø D9 Ø3  74Ø            JSR  RWTS           RWTS-Aufruf
ØE26: A9 ØØ     741            LDA  #$ØØ
ØE28: 85 48     742            STA  PREG           Bug-Fix
ØE2A: 6Ø        743            RTS
                744  *
ØE2B: 2Ø E3 Ø3  745  GETIOB    JSR  RWTSLOC        lädt Y-A
ØE2E: 84 Ø7     746            STY  IOBPNT
ØE3Ø: 85 Ø8     747            STA  IOBPNT+1
ØE32: 6Ø        748            RTS
```

1584 Bytes

Designer

Zeichensatz-Generator in Apple-Pascal

von Gabor Herr



Wenn Sie zu den Programmierern gehören, die vergeblich versucht haben, in Apple-Pascal im Grafikmodus einen deutschen Text mit Umlauten auf den Bildschirm zu bringen, oder den mitgelieferten Zeichensatz einfach langweilig finden und sich künstlerisch betätigen möchten, dann ist dieses Programm genau richtig für Sie.

1. Zeichendarstellung in Pascal

Im Gegensatz zu Applesoft-BASIC ist es in Pascal möglich, Strings oder Texte mit Hilfe der beiden Befehle „Wchar" und „Wstring" im Grafikmodus auf dem Bildschirm auszugeben. Wie funktionieren nun diese beiden Prozeduren?

In der Turtlegraphics-Unit wurde die Anweisung „Drawblock" implementiert, mit deren Hilfe sich eine Punktmatrix auf dem Grafikbildschirm darstellen läßt. Wurde z.B. eine 10x10-Punkte große Matrix in einem Feld, das folgendermaßen deklariert wurde:

Bild:
Packed Array (1..10,1..10) of Boolean
abgelegt, so kann es mit

Drawblock (Bild, 2, 0, 0, 10, 10, 5, 20, 10)
auf den Hires-Bildschirm ausgegeben werden. Die Matrixpunkte werden vom Pascal-System als Bits in Bytes zusammengefaßt und in der Variablen „Bild" abgespeichert. Beim Aufrufen der Prozedur „Drawblock" werden diese Bytes zeilenweise in den Grafikspeicher kopiert. Deshalb muß als zweiter Parameter die Anzahl der Bytes, die zur Darstellung einer Zeile nötig sind, angegeben werden. Die für den Kopiervorgang benötigten Daten werden durch die folgenden Parameter bestimmt:

Drawblock (Matrix, Bytes, xs, ys, w, h, x0, y0, Mode)

Matrix: Variable, die die Punktmatrix enthält
Bytes: Anzahl der Bytes ((w+15) div 16) * 2
xs, ys: Anzahl der Punkte, die bei der Darstellung in x- bzw. y-Richtung übersprungen werden sollen
w, h: Breite und Höhe der Matrix
x0, y0: Koordinaten der Bildschirmposition (linke untere Ecke der Matrix)
Mode: Modus wie bei Chartype (z.B. 10 für „Kopiere Array")

Bei der Darstellung von Texten greifen „Wchar" und „Wstring" auf diese Prozedur zurück. Sie verwenden für jedes Zeichen eine 8x8-Punktmatrix. Der Zeichensatz enthält 128 Zeichen und ist nach dem bekannten ASCII-Code aufgebaut. Es wurden lediglich die Ctrl-Zeichen durch Grafiksymbole (z.B.: Apfelzeichen, Schachfiguren usw.) ersetzt. Die Zeichenmuster sind in einem Feld zusammengefaßt und in einer Datei unter dem Namen „SYSTEM.CHARSET" auf der Diskette „APPLE1" abgelegt. Bei der Initialisierung der Turtlegraphics-Unit wird der Zeichensatz in den Arbeitsspeicher geladen, wo er während der Programmausführung auch bleibt. Mit Hilfe des „Designers" ist es nun möglich, diese oder eine genauso aufgebaute Datei zu laden, die einzelnen Zeichen zu editieren und sie dann wieder abzuspeichern.

2. Bedienung des Programms

Das Programm ist menügesteuert, ähnlich wie das Pascal-System. Bei der Eingabe der folgenden Kommandos dürfen Sie kleine und große Buchstaben verwenden:

L(oad: Laden eines Zeichensatzes von der Diskette. Ist die gesuchte Datei nicht vorhanden oder hat sie nicht das passende Format, so wird die Fehlermeldung „Datei fehlt" ausgegeben.
S(ave: Speichern des aktuellen Zeichensatzes auf der Diskette. Eine schon bestehende Datei mit dem gleichen Namen wird überschrieben. Deshalb geben Sie nur dann „SYSTEM.CHARSET" ein, wenn davon bereits eine Sicherheitskopie existiert.
N(eu: Löschen des Speichers. Nach der Eingabe dieses Kommandos werden Sie gefragt, ob nur das angezeigte Zeichen (= C(har), oder der ganze Zeichensatz (= A(lle) gelöscht werden soll.
V(erify: Das aktuelle Zeichen wird neben der Matrix auch in Originalgröße dargestellt.
Q(uit: Beenden des Programms
+, -: Das folgende bzw. vorangehende Zeichen wird zum Editieren auf dem Bildschirm angezeigt. Diesem Kommando kann ein Faktor (2-127) vorangestellt werden. Die Eingabe „30+" würde beispielsweise die Zeichenposition um 30 Zeichen vorrücken.
Leertaste: Mit Hilfe der Leertaste kann ein Punkt in der Matrix gesetzt bzw. gelöscht werden.

Die Ebenen Save, Load und Neu können bei falschen Eingaben mit der „ESC"-Taste verlassen werden. Die Tastenbelegung für die Kursorsteuerung ist durch Konstanten (cu_l, cu_r... usw.) im Deklara-

tionsteil festgelegt. Die im Listing abgedruckten Zahlen beziehen sich auf einen Apple IIe; für andere Apple-Typen können sie leicht abgeändert werden.

3. Beispiel

Nach dem Starten des Programms erscheint auf dem Monitor ein großes Quadrat für die Darstellung der Matrix und eine Zeile mit den Kommandos. Gibt man die Anweisung „L" für Load und dann „*SYSTEM.CHARSET" ein, so wird der Standard-Zeichensatz geladen. Nach dem Drücken der „+"-Taste erscheint das erste Zeichen, hier das „Apple"-Symbol, vergrößert auf dem Bildschirm. Mit Hilfe der Leertaste ist es jetzt möglich, den Punkt auf der Kursorposition zu setzen bzw. zu löschen. Um das neu erstellte Zeichen in Originalgröße zu sehen, muß das Kommando „Verify" eingegeben werden. Jede Änderung wird direkt in den Speicher geschrieben, so daß man mit „Save" den aktuellen Zeichensatz jederzeit auf der Diskette abspeichern kann.

Kurzhinweise

1. Zweck:
Entwurf neuer und Änderung alter Zeichensätze
2. Konfiguration:
Apple II+/e/c; Apple-Pascal 1.1 oder 1.2
3. Test:
E(xecute DESIGNER.CODE
4. Sammeldisk:
DESIGNER.TEXT
(Quelltext als DOS-3.3-Textfile, der mit GETDOS konvertiert und dann als DESIGNER.CODE compiliert werden muß.)

DESIGNER.TEXT

```
Unter Apple-Pascal 1.1 oder 1.2
über C(ompile als DESIGNER.CODE
compilieren.
Dann zu Übungszwecken über
E(execute DESIGNER.CODE starten
und über L(oad und
Welche Datei laden? "#4:SYSTEM.CHARSET"
einlesen.

(*******************************)
(*  Apple Pascal Designer 1.0  *)
(*                             *)
(* Programm zur Aenderung von  *)
(* Zeichensaetzen, die bei     *)
(* den Prozeduren 'Wchar' und  *)
(* 'Wstring' verwendet werden. *)
(*******************************)

{$C Gabor Herr 1984}
{$S+}

program designer;

   uses turtlegraphics;

   const x_anf=100;
         y_anf= 20; {Bildschirmposition der Matrix}
         breite=10; {Groesse eines Matrixpunktes  }

         {Kursortasten:}
         cu_l=8 ;{ links }
         cu_r=21;{ rechts}
         cu_u=10;{ oben  }
         cu_d=11;{ unten }

         {Steuertasten:}
         esc=27; { escape}
         bs =8 ; { backspace}

   type  tchartable=packed array[0..63] of boolean;
         tcharset  =array[0..127] of tchartable;

   var   charfile :file of tcharset;
         charset  :tcharset;
         chartable:tchartable;
         index,zeile,spalte,indiff,faktor:integer;

  procedure writes(pattern:string;x,y:integer);
     {schreibt String auf der angegebenen Position}
     begin
        pencolor(none);
        moveto(x,y);
        wstring(pattern)
     end;

  procedure cleartxt;
     {loescht Fenster auf dem Bildschirm}
     begin
        viewport(0,279,145,168);
        fillscreen(black);
        viewport(0,279,0,191)
     end;

  procedure clearch;
     {loescht Matrix vom Bildschirm}
     begin
        viewport(99,180,19,100);
        fillscreen(black);
        viewport(0,279,0,191)
     end;
```

```
            if key=chr(bs)
            {bei <bs> zuletzt eingegebenes Zeichen loeschen}
               then
                  begin
                     if idx>1 then
                        begin
                           idx:=idx-1;
                           move(7);
                           wchar(' ');
                           move(7)
                        end;
                  end
               else
                  {sonst Echo ausgeben}
                  begin
                     wchar(key);
                     inword[idx]:=key;
                     idx:=idx+1
                  end;
         until eoln(keyboard);
         inword:=copy(inword,1,idx-1);
      end;

   procedure plotpoint(xpos,ypos:integer;mode:boolean);
      {setzt einen Punkt in die Matrix auf dem Bildschirm}
      var color:screencolor;
      begin
         if mode then color:=white
                 else color:=black;
         xpos:=xpos*breite+x_anf;
         ypos:=ypos*breite+y_anf;
         viewport(xpos+1,xpos+breite-2,ypos+1,ypos+breite-2);
         fillscreen(color);
         viewport(0,279,0,191)
      end;

   procedure writechar(nr:integer);
      {zeichnet einen Charakter vergroessert
        und in Originalgroesse}
      var zle,spt:integer;
          nbr:string;
      begin
         for zle:=0 to 7 do
            begin
               for spt:=0 to 7 do
                  begin
                     if chartable[zle*8+spt]
                        then plotpoint(spt,zle,true);
                  end;
            end;
         str(nr,nbr);writes(concat(nbr,' '),66,90);
         drawblock(chartable,1,0,0,8,8,201,91,10)
      end;

   procedure input(var inword:string;x,y:integer);
      {String-Eingabe mit Echo im Grafikmodus}
      var idx:integer;
          key:char;
      begin
         idx:=1;fillchar(inword,40,chr(32));
         pencolor(none);moveto(x,y);turnto(180);
         repeat
            read(keyboard,key);
            if key=chr(esc) then
                              {bei <esc> Eingabe abbrechen}
                              begin
                                 inword:='';
                                 exit(input)
                              end;

   procedure savechset;
      {speichert den aktuellen Zeichensatz auf Diskette}
      var filename:string;
```

PEEKER Börse

Gelegenheitsanzeigen

Sie können unter dieser Rubrik zu einem besonders günstigen Preis

- Ihre Hardware und Software verkaufen
- Ihre Hard- und Software suchen
- Kontakte knüpfen und vieles mehr

Musteranzeige privat

1 Druckzeile à 32 Buchstaben nur DM 5,50 zuzügl. ges. MwSt. Mindestens 2 Druckzeilen

Beispiel:

> Verkaufe neuwertigen Typenraddrucker mit Apple-Interface. Preis auf Anfrage. Tel. 007

nur DM 18,81 inkl. MwSt.

Musteranzeige gewerblich

1 Druckzeile à 32 Buchstaben nur DM 11,– zuzügl. ges. MwSt. Mindestens 2 Druckzeilen

Beispiel:

> Neu im Angebot: Professionelle, separate Tastatur für Apple II plus 16 Funktionstasten und separatem Ziffernblock.
> Fa. Keyboard & Co.

nur DM 62,70 inkl. MwSt.

PEEKER

Peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

Bitte veröffentlichen Sie in der nächsterreichbaren Ausgabe nachstehenden Text unter folgender Rubrik:

☐ suche Hardware ☐ suche Software ☐ Verschiedenes
☐ biete Hardware ☐ biete Software ☐ Chiffre

Bitte jeweils 32 Buchstaben pro Zeile – einschließlich Satzzeichen und Wortzwischenräume. Bitte Absender nicht vergessen. Mindestens 2 Zeilen. Chiffregebühr DM 6,– zuzügl. MwSt.

PEEKER

Produkt-Karte

Zu der im **Peeker**, Heft _____, Seite _____ erschienenen

Anzeige über _____

_____

bitte ich um detaillierte Information.

Ich wünsche ☐ Prospekt, Datenblatt ☐ Preisliste ☐ schriftliches Angebot ☐ tel. Rückruf

Menge	Produkt und Bestellnummer	à DM	gesamt DM

gebe ich nebenstehende Bestellung unter Anerkennung Ihrer in der Anzeige genannten Liefer- und Zahlungsbedingungen auf.

_____

Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

PEEKER

Info-Karte

Schreiben Sie uns, wenn Sie technische Fragen zu Aufsätzen im **Peeker** haben. Beachten Sie jedoch, daß die Redaktion keine Auskünfte über Bezugsquellen erteilt:

_____

_____

_____

_____

_____

_____

Peeker-Börse

Vorname, Name

Firma

Straße

Wohnort

PLZ/Ort

Bitte veröffentlichen Sie den umstehenden Text von _____ Zeilen à _____ DM in der nächsterreichbaren Ausgabe vom **Peeker**

Bei Angeboten: Ich bestätige, daß ich alle Rechte an den angebotenen Sachen besitze

Datum Unterschrift

POSTKARTE

Peeker-Börse
Anzeigen-Service
Dr. Alfred Hüthig Verlag
Postfach 10 28 69
6900 Heidelberg 1

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Anschrift der Firma angeben, bei der Sie bestellen bzw. von der Sie Informationen wünschen

POSTKARTE

Inserent

Straße

PLZ/Ort

Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker
Redaktion
Dr. Alfred Hüthig Verlag
Postfach 10 28 69
6900 Heidelberg 1

Produkt-Karte

Wünschen Sie weitere Informationen zu einer der im Heft erschienenen Anzeigen?

Nichts einfacher als das. Produkt-Karte ausfüllen, frankieren und an den Inserenten (nicht an die Peeker-Redaktion senden.

Vorher aber nicht vergessen:
Kreuzen Sie an, welchen Informationswunsch Sie haben.

Damit erleichtern Sie dem Hersteller eine gezielte Beantwortung Ihrer Anfrage

Zum Schluß tragen Sie auf der Rückseite die genaue Anschrift des Inserenten und Ihren Absender ein.

PEEKER

```
   {Die Ein- und Ausgabekontrollen
     werden unterdrueckt, um zu
     verhindern, dass beim Auftreten
     eines Fehlers das Programm ab-
     stuerzt.}
   {$i-}

   begin
      writes('Welche Datei speichern?',Ø,16Ø);
      input(filename,Ø,152);
      cleartxt;
      if filename='' then exit(savechset);
      rewrite(charfile,filename);
      if ioresult<>Ø then
         {Fehlerbehandlung}
         begin
            writes('I/O-Fehler',Ø,16Ø);
            writes('Return eingeben',Ø,152);
            readln;cleartxt
         end;
      charfile↑:=charset;
      put(charfile);
      close(charfile,lock)
   end;

procedure loadchset;
   {holt Zeichensatz von der Diskette}
   var filename:string;
       I, J : integer;

   begin
      writes('Welche Datei laden?',Ø,16Ø);
      input(filename,Ø,152);
      cleartxt;
      if filename='' then exit(loadchset);
      reset(charfile,filename);
      if ioresult<>Ø then
         {Fehlerbehandlung}
         begin
            writes('Datei existiert nicht',Ø,16Ø);
            writes('Return eingeben',Ø,152);
            readln;cleartxt
         end;
      charset:=charfile↑;
      close(charfile);
      index:=Ø;
      chartable:=charset[index];
      for I := Ø to 3 do
        for J := Ø to 31 do
          drawblock(charset[I*32+J],1,Ø,Ø,8,8,J*8,135-I*9,1Ø)
   end;
   {$i+} {Kontrolle wieder einschalten}

procedure newset;
   {loescht den aktuellen Zeichensatz}
   begin
      fillchar(charset,sizeof(charset),chr(Ø));
      index:=Ø;
      chartable:=charset[index]
   end;

procedure newchar;
   {loescht das angezeigte Zeichen}
   begin
      fillchar(chartable,sizeof(chartable),chr(Ø))
   end;

procedure cursor(x,y:integer);
   {Kursorposition anzeigen}
   begin
      chartype(6);
      moveto(x*breite+x_anf+1,y*breite+y_anf+1);
      wchar('*');chartype(1Ø)
   end;

procedure new;
   {loeschen}
   var key:char;
   begin
      writes('Neuer C(har oder A(lle?',Ø,16Ø);
      repeat
         read(keyboard,key)
      until (key in ['c','C','a','A']) or (key=chr(esc));
      if key<>chr(esc) then
         begin
            if key in ['c','C']
               then newchar
               else newset;
         end;
      cleartxt
   end;
```

```
procedure invert;
   {Punkt invertieren}
   begin
      if chartable[zeile*8+spalte]
         then
            chartable[zeile*8+spalte]:=false
         else
            chartable[zeile*8+spalte]:=true;
      plotpoint(spalte,zeile,chartable[zeile*8+spalte]);
      drawblock(chartable,1,Ø,Ø,8,8,2Ø1,91,1Ø);
      cursor(spalte,zeile)
   end;

procedure initialize;
  {Variablen initialisieren und Bildschirm aufbauen}
  begin
    newset;
    index:=Ø;indiff:=Ø;spalte:=Ø;zeile:=Ø;
    initturtle;
    moveto(Ø,18Ø);wchar(chr(1));
    writes('designer 1.Ø *N(eu,S(ave,L(oad,Q(uit',16,18Ø);
    writes('v(erify,[-,+] next char,[sp] invert bit*',Ø,17Ø);
    moveto(Ø,169);pencolor(white);moveto(279,169);
    pencolor(none);moveto(Ø,144);
    pencolor(white);moveto(279,144);
    pencolor(none);moveto(Ø,1Ø6);
    pencolor(white);moveto(279,1Ø6);
    writes('char nr:Ø',1Ø,9Ø);
    moveto(199,89);turnto(Ø);pencolor(white);
    move(11);turn(9Ø);move(11);turn(9Ø);
    move(11);turn(9Ø);move(11);
    pencolor(none);
    viewport(97,182,17,1Ø2);fillscreen(white);
    clearch
  end;

procedure edit;
   {Das eigentliche Hauptprogramm}
   var key:char;

   begin
      repeat
        cursor(spalte,zeile);
        repeat
           read(keyboard,key)
        until key in ['n','N','s','S','l','L','q','Q',
                      'v','V',' ','+','=','-','_',
                      'Ø'..'9', chr(cu_l), chr(cu_r),
                      chr(cu_u),chr(cu_d)];
        case key of
           'n','N':new;
           's','S':savechset;
           'l','L':loadchset;
           '+','=':if faktor=Ø then indiff:= 1
                                else indiff:=faktor;
           '-','_':if faktor=Ø then indiff:=-1
                                else indiff:=-faktor;
           ' '    :invert;
           end;
        cursor(spalte,zeile);
        if key in [' ', 'Ø'..'9', chr(cu_l), chr(cu_r),
                   chr(cu_u),chr(cu_d)]
           then
              begin
                 case ord (key) of
                   cu_l : if spalte>Ø then spalte:=spalte-1;
                   cu_r : if spalte<7 then spalte:=spalte+1;
                   cu_u : if zeile>Ø  then zeile :=zeile-1;
                   cu_d : if zeile<7  then zeile :=zeile+1;
                 end;
                 if key in ['Ø'..'9'] then
                    {Voreinstellungsfaktor}
                       faktor:=faktor*1Ø+ord(key)-48
              end
           else
              begin
                 charset[index]:=chartable;
                 drawblock(chartable,1,Ø,Ø,8,8,index mod 32*8,
                           135-index div 32*9,1Ø);
                 index:=index+indiff;
                 indiff:=Ø;faktor:=Ø;
                 if index<Ø   then index:=Ø;
                 if index>127 then index:=Ø;
                 chartable:=charset[index];
                 clearch;writechar(index);
              end;
     until key in ['q','Q'];
  end;

{Hauptprogramm}
begin
   initialize;
   edit
end.
```

READPAS

Konvertierung von UCSD- in Turbo-Pascal-Textfiles

von Michael Erperstorfer

Wer bislang unter UCSD-Pascal bzw. Apple-Pascal 1.1 oder 1.2 gearbeitet hat und nunmehr Programme unter Turbo-Pascal entwickeln will, wird sicherlich einen Teil seiner alten UCSD-Programme übernehmen wollen. Allerdings kennt UCSD-Pascal zahlreiche Befehle, die Turbo-Pascal nicht kennt und umgekehrt. Ferner sind diverse Feinheiten zu beachten, so daß es mit der Konvertierung des UCSD-Pascal-Textfiles allein nicht getan ist.

Das nachfolgende Turbo-Pascal-Programm **READPAS** läuft sowohl unter Turbo 2.0 als auch 3.0. Es ist jedoch darauf zu achten, daß der Quelltext auf Diskette compiliert werden muß, nachdem zuvor die Endadresse (nicht die Startadresse!) auf $C7FF gesetzt worden ist. Verfahren Sie im einzelnen wie folgt:

1. Turbo-Pascal 2.0 oder 3.0 starten.
2. Quelltext eingeben und unter READPAS.PAS abspeichern.
3. O(ptionen und dann C(om-File wählen und zum Schluß E(nd-Address auf $C7FF setzen.
4. Nunmehr vom Hauptmenü aus C(ompilieren.

Die entstandene Objektcode-Datei starten Sie dann unter dem CP/M-Betriebssystem wie eine übliche COM-Datei mit
>READPAS
Das Programm setzt zwei Laufwerke voraus. Die CP/M- bzw. Turbo-Pascal-Arbeitsdiskette legen Sie zweckmäßigerweise in Laufwerk 1 (A; #4) und die Apple-Pascal-Diskette in Laufwerk 2 (B; #5). Nach dem Start von READPAS.COM können Sie sich zunächst über Menü-Option 2 das UCSD-Pascal-Directory ansehen. Danach brauchen Sie nur noch den Namen des gewünschten UCSD-Textfiles und darauf den ggf. gekürzten Turbo-Textfile-Namen (mit Suffix „.PAS") einzugeben. Die konvertierte Datei läßt sich direkt in den Turbo-Pascal-Editor einlesen.

READPAS

```
PROGRAM READPASCAL; {von Michael Erperstorfer}

CONST BUFFSTART=$C800; {START DES 128 BYTE LESEPUFFER}

TYPE
  FILETYP=RECORD
   FNAME:STRING[15];
   VON:0..280;
   BIS:0..280;
   FTYP:STRING[4];
   EOFBYTE:0..511
  END;

VAR
  BUFFER:ARRAY[0..127] OF BYTE ABSOLUTE BUFFSTART; {LESEPUFFER}
  BLOCK:ARRAY[0..511] OF BYTE; {PASCAL-BLOCK}
  DIRBLOCK:ARRAY[0..2047] OF BYTE; {4 BLOCKS FUER DIRECTORY}
  PASBUFF:ARRAY[0..30719] OF BYTE; {PUFFER FUER PASCAL TEXT-FILE}
  TAB:ARRAY[0..7,1..4] OF 0..31; {LOOK-UP TABELLE}
  EINTRAG:ARRAY[1..77] OF FILETYP; {MAX 77 FILES IN PASCAL}
  OUT:TEXT;
  VOLNAME:STRING[15];
  BEZUGSLW,FILEZAHL,BLZAHL:INTEGER;
  LWP,LWC:CHAR;

PROCEDURE TABINIT; {SEKTOR/BLOCK-UMWANDLUNG PASCAL-CP/M}
 BEGIN
  TAB[0,1]:= 0; TAB[0,2]:= 1; TAB[0,3]:=12; TAB[0,4]:=13;
  TAB[1,1]:=24; TAB[1,2]:=25; TAB[1,3]:= 4; TAB[1,4]:= 5;
  TAB[2,1]:=16; TAB[2,2]:=17; TAB[2,3]:=28; TAB[2,4]:=29;
  TAB[3,1]:= 8; TAB[3,2]:= 9; TAB[3,3]:=20; TAB[3,4]:=21;
  TAB[4,1]:=22; TAB[4,2]:=23; TAB[4,3]:= 2; TAB[4,4]:= 3;
  TAB[5,1]:=14; TAB[5,2]:=15; TAB[5,3]:=26; TAB[5,4]:=27;
  TAB[6,1]:= 6; TAB[6,2]:= 7; TAB[6,3]:=18; TAB[6,4]:=19;
  TAB[7,1]:=30; TAB[7,2]:=31; TAB[7,3]:=10; TAB[7,4]:=11
 END;

PROCEDURE READBLOCK(BLNUM:INTEGER);
 VAR SPURNUM,TABINDEX,I,ERROR:INTEGER;

 PROCEDURE FILLBLOCK(NUM:INTEGER); {FUELLT PASCAL-BLOCKPUFFER}
  VAR J,BLINDEX:INTEGER; {MIT CP/M SEKTOR (4 MAL)}
  BEGIN
   BLINDEX:=(NUM-1)*128;
   FOR J:=0 TO 127 DO
    BEGIN
     BLOCK[BLINDEX]:=BUFFER[J];
     BLINDEX:=BLINDEX+1
    END
  END;

 BEGIN
  SPURNUM:=BLNUM DIV 8;
  TABINDEX:=BLNUM MOD 8;
  FOR I:=1 TO 4 DO
   BEGIN
    BIOS(11,BUFFSTART); {SETZT BEGINN DES LESEPUFFERS}
    BIOS(7);      {SETZT KOPF AUF SPUR 0}
    BIOS(9,SPURNUM); {SETZT SPURNUMMER}
    BIOS(10,TAB[TABINDEX,I]); {SETZT SEKTORNUMMER}
    BIOS(8,ORD(LWP)-65); {SETZT LAUFWERK}
    ERROR:=BIOS(12); {LIEST SEKTOR IN LESEPUFFER}
    IF ERROR<>0 THEN
     BEGIN
      WRITELN(CHR(7),'I/O FEHLER IN LAUFWERK ',LWP,'.'); WRITELN;
      WRITE('CP/M-DISKETTE EINLEGEN UND <RETURN> DRUECKEN ');
      READLN;
      BDOS(0)
     END;
    FILLBLOCK(I);
    WRITE('.')
   END
 END;

PROCEDURE READDIRECTORY;
 VAR I,J:INTEGER;

 PROCEDURE GETFILES;
  VAR K,INDEX,TYPNUM:INTEGER;
  BEGIN
   INDEX:=26;
   FILEZAHL:=0;
   REPEAT
    IF DIRBLOCK[INDEX+6]>0 THEN
     BEGIN
      FILEZAHL:=FILEZAHL+1;
      WITH EINTRAG[FILEZAHL] DO
       BEGIN
        VON:=DIRBLOCK[INDEX]+DIRBLOCK[INDEX+1]*256;
        BIS:=DIRBLOCK[INDEX+2]+DIRBLOCK[INDEX+3]*256;
        EOFBYTE:=DIRBLOCK[INDEX+22]+DIRBLOCK[INDEX+23]*256-1;
        TYPNUM:=DIRBLOCK[INDEX+4];
        CASE TYPNUM OF
         2:FTYP:='CODE';
         3:FTYP:='TEXT';
         5:FTYP:='DATA'
        END;
        FNAME[0]:=CHR(DIRBLOCK[INDEX+6]);
        FOR K:=1 TO DIRBLOCK[INDEX+6] DO
         FNAME[K]:=CHR(DIRBLOCK[6+INDEX+K])
       END
     END;
    INDEX:=INDEX+26
   UNTIL INDEX>2021
  END;

 BEGIN
  CLRSCR;
  WRITE('Das Directory wird gelesen');
  FOR I:=2 TO 5 DO
```

```
    BEGIN
     READBLOCK(I);
     FOR J:=0 TO 511 DO DIRBLOCK[(I-2)*512+J]:=BLOCK[J]
    END;
   VOLNAME[0]:=CHR(DIRBLOCK[6]);
   FOR I:=1 TO DIRBLOCK[6] DO VOLNAME[I]:=CHR(DIRBLOCK[6+I]);
   GETFILES
  END;

 PROCEDURE PRINTDIRECTORY;
  VAR I,ZAEHLER:INTEGER;
  BEGIN
   IF FILEZAHL>0 THEN
    BEGIN
     CLRSCR;
     ZAEHLER:=0;
     WRITELN('Directory der Pascal-Diskette ',VOLNAME,':');
     WRITELN;
     WRITELN('     Filename  Laenge in Blocks  Filetyp');
     WRITELN('-----------------------',
             '-----------------------------------');
     FOR I:=1 TO FILEZAHL DO WITH EINTRAG[I] DO
      BEGIN
       WRITELN(FNAME:20,(BIS-VON):10,'      ',FTYP,'-FILE');
       ZAEHLER:=ZAEHLER+1;
       IF (ZAEHLER=18) OR (I=FILEZAHL) THEN
        BEGIN
         WRITELN;
         WRITE('Weiter mit <return> ');
         READLN;
         ZAEHLER:=0;
         CLRSCR
        END
      END
    END
  END;

 PROCEDURE TRANSFER;
  VAR I,J,INDEX,ANFANG,ENDE:INTEGER;
    PASNAME,CPMNAME:STRING[15];
    TABFLAG:BOOLEAN;
    EIN:CHAR;

  FUNCTION GETNAME:BOOLEAN;
   BEGIN
    CLRSCR;
    WRITE('Name des Pascal-Textfiles: ');
    READLN(PASNAME);
    IF POS('.TEXT',PASNAME)=0
     THEN PASNAME:=CONCAT(PASNAME,'.TEXT');
    GETNAME:=FALSE;
    FOR I:=1 TO FILEZAHL DO
     IF EINTRAG[I].FNAME=PASNAME THEN
      BEGIN
       GETNAME:=TRUE;
       INDEX:=I
      END
   END;

  PROCEDURE WRITEFILE;
   VAR IND,I:INTEGER;
   BEGIN
    IND:=0;
    FOR I:=ANFANG TO ENDE DO
     BEGIN
      READBLOCK(I);
      FOR J:=0 TO 511 DO
       BEGIN
        PASBUFF[IND]:=BLOCK[J];
        IND:=IND+1
       END
     END;
    READBLOCK(ENDE+1);
    FOR J:=0 TO EINTRAG[INDEX].EOFBYTE DO
     BEGIN
      PASBUFF[IND]:=BLOCK[J];
      IND:=IND+1
     END;
    BIOS(8,BEZUGSLW);
    IND:=IND-1;
    TABFLAG:=FALSE;
    WRITELN; WRITE('Name des CP/M-Files (ohne Laufwerksangabe):');
    READLN(CPMNAME);
    CPMNAME:=CONCAT(LWC,':',CPMNAME);
    ASSIGN(OUT,CPMNAME);
    REWRITE(OUT);
    FOR I:=0 TO IND DO
     IF TABFLAG THEN
      BEGIN
       FOR J:=1 TO PASBUFF[I]-32 DO WRITE(OUT,' '); {LEERZEICHEN}
       TABFLAG:=FALSE
      END
     ELSE
      BEGIN
       CASE PASBUFF[I] OF
        13:WRITE(OUT,CHR(13),CHR(10)); {LINEFEED ANFUEGEN}
        16:TABFLAG:=TRUE; {NAECHSTES ZEICHEN: EINRUECK-WERT}
        32..126:WRITE(OUT,CHR(PASBUFF[I])) {NUR LESBARE ZEICHEN}
       END;
       IF I MOD 128=0 THEN WRITE('.')
      END;
    CLOSE(OUT)
   END;
  BEGIN
   CLRSCR;
   WRITELN('Pascal-Diskette in das Laufwerk ',
           LWP,' einlegen.'); WRITELN;
   WRITE('F)ertig oder zurueck zum M)enu ? ');
   REPEAT
    READ(KBD,EIN); EIN:=UPCASE(EIN)
   UNTIL EIN IN ['F','M'];
   IF EIN='F' THEN
    BEGIN
     READDIRECTORY;
     IF NOT GETNAME THEN
      BEGIN
       WRITE(PASNAME,
             ' nicht gefunden ! Weiter mit <return> ');
       READLN;
       PRINTDIRECTORY
      END
     ELSE
      BEGIN
       ANFANG:=EINTRAG[INDEX].VON+2;
       ENDE:=EINTRAG[INDEX].BIS-2;
       IF (ENDE-ANFANG)>58
        THEN WRITELN(CHR(7),'Programm zu gross!')
       ELSE WRITEFILE
      END
    END
  END;
 PROCEDURE MENU;
  VAR AUSWAHL:CHAR;
    ENDE:BOOLEAN;
  BEGIN
   ENDE:=FALSE;
   LWP:='B'; {DEFAULT P-SYSTEM: LAUFWERK C}
   LWC:='A'; {DEFAULT CP/M: LAUFWERK D}
   REPEAT
    CLRSCR;
    GOTOXY(28,1); WRITELN('R E A D P A S C A L'); WRITELN;
    WRITELN('              von Michael Erperstorfer'); WRITELN;
    WRITELN('         Programm zum Lesen von UCSD-Textfiles');
    WRITELN;
    WRITELN('1...Laufwerk-Daten aendern (CP/M: ',
            LWC,' Pascal: ',LWP,')');
    WRITELN;
    WRITELN('2...Directory der Pascal-Diskette'); WRITELN;
    WRITELN('3...Pascal-Texfile nach CP/M uebertragen'); WRITELN;
    WRITELN('4...E N D E'); WRITELN; WRITELN;
    WRITE('Auswahl: ');
    REPEAT
     READ(KBD,AUSWAHL); WRITELN(AUSWAHL)
    UNTIL AUSWAHL IN ['1'..'4'];
    CASE AUSWAHL OF
     '1':BEGIN
         CLRSCR;
         WRITE('Laufwerk fuer die CP/M-Diskette (<A>,<B>): ');
         REPEAT
          READ(KBD,LWC); WRITELN(LWC);
          LWC:=UPCASE(LWC)
         UNTIL LWC IN ['A','B'];
         WRITE('Laufwerk fuer die Pascal-Diskette (<A>,<B>): ');
         REPEAT
          READ(KBD,LWP); WRITELN(LWP);
          LWP:=UPCASE(LWP)
         UNTIL LWP IN ['A','B']
        END;
     '2':BEGIN
         READDIRECTORY;
         PRINTDIRECTORY
        END;
     '3':TRANSFER;
     '4':ENDE:=TRUE
    END
   UNTIL ENDE
  END;
 BEGIN
  BEZUGSLW:=BDOS(25);
  TABINIT;
  MENU;
  BIOS(8,BEZUGSLW)
 END.
```

Kyan-Pascal

Ein vollcompilierendes 6502-Pascal

Erfahrungsbericht von Herbert Pohl

Hinweis: Der nachfolgende Erfahrungsbericht basiert noch auf den Kyan-Pascal-Versionen 1.0 und 1.1. Die verbesserte Version 1.2 wurde am 14.08.1985 ausgeliefert.

Was erwarten Sie, wenn Sie das Pascal-Programm **GENAU** unter UCSD-Pascal oder Turbo-Pascal auf einem Apple IIe laufen lassen? Unter Turbo-Pascal (V1.0 oder V2.0) wird der Rechner nach etwa 53s jammern, unter UCSD-Pascal nach etwa 144s. Ohne Ausgabe der Rechenergebnisse auf dem Bildschirm sind es 1,8s bzw. 3,5s, vom ersten Piepsen des Lautsprechers an gerechnet.
Unter Kyan-Pascal, Version 1.0 und 1.1, das seit einiger Zeit für den Apple II erhältlich ist, frohlockt der Rechner nach 24 bzw. 5s. Unter diesem Pascal wird also im Gegensatz zu den beiden anderen Pascal-Implementierungen bei den aufeinanderfolgenden Subtraktionen exakt der Wert 0 erreicht.
Begründet sein muß dies darin, daß Turbo- und UCSD-Pascal Real-Zahlen als Dualzahlen darstellen, während Kyan-Pascal offenbar die BCD-Darstellung benutzt. Letzteres ist im mit dem System mitgelieferten Tutorial Manual allerdings nicht dokumentiert, jedoch sprechen die im obigen Programm erreichte Genauigkeit, der durch Real-Zahlen darstellbare Zahlenbereich (betragsmäßig von 1.0E-99 bis 9.999999999999E+99, 13 geltende Ziffern) und der zur Darstellung benötigte Speicherplatz von acht Bytes hierfür.

1. Lieferumfang

Aufgrund einer Anzeige in der Zeitschrift „Incider" habe ich das Kyan-Pascal-System Anfang Mai in den USA bestellt und innerhalb von vierzehn Tagen für knapp 60.00 Dollar erhalten; inzwischen beträgt der Preis einschließlich Versand für Europa knapp 82.00 Dollar. Kyan-Pascal soll übrigens auch für den Atari 800 XL und den Commodore C64 erhältlich sein.
Geliefert wurden eine Pascal-System-Diskette und ein Tutorial Manual, wobei die nähere Untersuchung des Systems das Vorhandensein einiger Fehler im Manual zeigte. Ansonsten ist dieses Anleitungsbuch recht geschickt aufgebaut und insbesondere auch für Anfänger geeignet, sofern sie einigermaßen Englisch beherrschen. Die Bedienung von Editor und Compiler wird anhand von Beispielen erläutert. Hierauf erfolgt eine kurze Einführung in die Programmierung mit Pascal und die Vorstellung der meisten der in diesem System eingebauten Möglichkeiten.

Auf der Diskette der neuesten Version 1.2 befinden sich die Dateien

PRODOS (ProDOS-System 1.1.1)
FILER (ProDOS-Kopierprogramm)
ED (Kyan-Pascal-Editor 40 Z/Z)
E80 (Kyan-Pascal-Editor 80 Z/Z)
PC (Kyan-Pascal-Compiler)
LIB (Kyan-Pascal-Library)
HELLO.SYSTEM (Startup-Programm und zugleich Hauptmenü)

sowie Hilfsprogramme für ProDOS (z.B. DIR für Directory-Anzeige), Include-Dateien (für Hires-Grafik und String-Verarbeitung) und Beispielprogramme in Text- und Codeform. Kyan-Pascal läuft auf dem Apple II unter dem Betriebssystem ProDOS, das ebenso wie der ProDOS-Filer mitgeliefert wird und nur mit dem Kyan-System zusammen benutzt werden darf. Inzwischen habe ich (am 09.07.85) von Kyan Software gratis (!) die Version 1.1 des Systems erhalten. Auch sonst bin ich bislang sehr zuvorkommend bedient worden.
Meine Untersuchungen stützen sich auf die Version 1.0, sofern nicht anders erwähnt. In **Tabelle 3** finden Sie die Entstehungsdaten wichtiger Dateien.

2. Kurzeinführung

Angenommen, man will das Programm GENAU eingeben, compilieren und laufen lassen. Hierzu ist die Kyan-Diskette oder eine entsprechende Arbeitsversion in Laufwerk 1 zu stecken und der Rechner anzuschalten. Zunächst wird ProDOS und dann das Hauptmenü-Programm PASCAL.SYSTEM (jetzt HELLO.SYSTEM) geladen, das sich auf dem Bildschirm mit
KYAN PASCAL (Version 1.X)
meldet, worauf das System auf die Eingabe eines Kommandos wartet, die mit der Return-Taste abzuschließen ist. Von diesem Hauptmenü aus kann man durch Eingabe des jeweiligen Programmnamens

1. den Editor (ED oder E80)
2. den Compiler (PC) oder
3. den compilierten Objektcode (z.B. GENAU.O)

starten. Danach kommt man stets in das Hauptmenü zurück, das man jedoch durch Eingabe eines *anderen* ProDOS-Systemprogramms (z.B. BASIC.SYSTEM) verlassen kann. (Anm.d.Red.: Das Hauptmenü-Miniprogramm PASCAL.SYSTEM, das ab Version 1.2 HELLO.SYSTEM heißt, ist wegen der System-Bit-Map-Konflikte mit ProDOS erforderlich. Beispielsweise kann man das BASIC.SYSTEM nicht direkt aus dem Kyan-Editor heraus, wohl aber über das HELLO.SYSTEM starten.)

2.1. ED/E80 = Editor

ED ruft den 40-Z/Z-Editor auf, einen Bildschirmeditor, dessen Steuerbefehle sehr an die des Turbo-Pascal-Editors erinnern. Zur Steuerung des Cursors sind sowohl die Pfeiltasten als auch ↑S, ↑D, ↑E und ↑X möglich. Außerdem gibt es Befehle, um den Cursor wortweise, seitenweise und an den Anfang oder das Ende der Datei zu bewegen. Ferner kann man einzelne Zeichen oder Zeilen löschen, Textblöcke markieren, löschen und verschieben. Weiter kennt der Editor einen mit der ESC-Taste aufzurufenden Modus, in dem der Text auf Diskette abgespeichert, Text von der Diskette gelesen und eingefügt, der Name der Datei (eigentlich der ProDOS-Pfadname) geändert und die Editorfunktionen des Suchens und Ersetzens aufgerufen werden können. Aus diesem Modus heraus wird der Editor auch verlassen. Mittels dieses ESC-Modus ist es ebenso möglich, Textdateien zu transferieren, indem man sie in den Editor lädt, den Pfadnamen ändert und die Datei mit dem neuen Pfadnamen abspeichert.
Nach seinem Aufruf fordert der Editor, der für seine Arbeit den gesamten Speicher von $0800 bis $BAFF benutzt, die Eingabe eines ProDOS-Pfadnamens an. Findet er die zugehörige Datei, wird sie eingelesen und auf dem Bildschirm angezeigt, wobei der Cursor am Textanfang steht. Ist die Datei nicht zu finden, erfolgt die Meldung „FILE NOT FOUND. PRESS ANY KEY".

Nach einem Tastendruck wird der Bildschirm gelöscht, und man kann seinen Text eingeben. Eine Status- oder Kommandozeile wie bei Turbo- oder UCSD-Pascal gibt es nicht. Jedoch kann jederzeit mit <ESC> ein Hilfsmenü aufgerufen werden, ohne den eigentlichen Text zu stören. Von diesem Hilfsmenü aus ist auch eine Aufstellung aller Editorbefehle aufrufbar, was wiederum für Anfänger sehr hilfreich ist.
In der Version 1.0 unterstützt der Editor keine 80-Zeichenkarte. Die eingegebene Zeichenfolge wird beim Erreichen eines Zeilenendes einfach in der nächsten Zeile des Bildschirmes fortgesetzt, so daß mehrere Zeilen auf dem Bildschirm eine logische Zeile im Text ausmachen. Logische Zeilen müssen durch Einfügen von <CR> abgeschlossen werden. Mit dem Kyan-System der Version 1.0 geschriebene Programme können jedoch die 80-Zeichenkarte ohne Einschränkungen benutzen.
Mit der Version 1.1 ist nun neben dem 40-Z/Z-Editor ED ein 80-Z/Z-Editor **E80** verfügbar, der auch eine Tabulator-Funktion besitzt.

2.2. PC = Pascal-Compiler

Mit **PC** wird der Compiler aufgerufen, der sich mit „PC>" meldet und die Eingabe des Namens des zu compilierenden Quelltextes erwartet. Nach Tippen des entsprechenden Pfadnamens, etwa
GENAU,
beginnt der Compiler seine Arbeit und erzeugt ein 6502-Maschinenprogramm, das ebenfalls auf Diskette abgelegt wird, und zwar unter demselben Pfadnamen wie der Quelltext, jedoch mit einem angehängten „.O", also etwa „GENAU.O". Die Compilierungszeiten sind mit denen des UCSD-Pascal vergleichbar, jedoch ist der Fortgang der Compilierung in der Regel nicht auf dem Bildschirm verfolgbar.
Beim Starten des Compilers kann man dem Pfadnamen der Pascal-Quelldatei eine oder mehrere Compileroptionen anfügen, die durch Bindestriche vom Pfadnamen und voneinander zu trennen sind.
-Ln erzeugt hierbei die Ausgabe des compilierten Programmes als Assemblerdatei über Steckplatz n,
-En lenkt die Fehlermeldungen auf Steckplatz n um, die sonst auf dem Bildschirm erscheinen, während
-O die Erzeugung des Objektcodes unterdrückt. Die Angabe des Steckplatzes eines Diskettenlaufwerkes führt allerdings zu einem Absturz des Compilers. Ich habe herausgefunden, daß die Eingabe dieser Optionen mit Großbuchstaben erfolgen muß und auch eine bestimmte Reihenfolge einzuhalten ist, wenn alles reibungslos ablaufen soll. Das Tutorial Manual weist hierauf nicht hin.
In der Version 1.1 sind die Bezeichnungen für die Optionen leicht geändert worden. Auch ist jetzt Kleinschreibung und eine beliebige Reihenfolge zulässig:
„-Sn" wählt das Peripheriegerät an, auf dem dann zumindest die Fehlermeldungen erscheinen, „-L" erzeugt die Ausgabe der Assemblerdatei auf dem angewählten Peripheriegerät. Die Option „-O" arbeitet wie in Version 1.0.
Mit Hilfe dieser Option ist eine reine Syntaxprüfung des Quelltextes möglich, die etwas weniger Zeit beansprucht als eine vollständige Compilierung. Da der Kyan-Compiler nach Auftreten eines Syntaxfehlers zwar die Erzeugung von Objektcode einstellt, aber die Syntaxprüfung ohne Pause weiter durchführt, empfiehlt es sich, die Fehlermeldungen auf einen Drucker zu lenken, wenn man nicht die Ausgabe mit ↑S unterbrechen will. Der Compilerlauf selbst kann übrigens jederzeit mit der Reset-Taste abgebrochen werden, ohne daß die Quelldatei verlorengeht.

2.3. RUN

Nach erfolgreicher Compilierung wird das Programm gestartet, indem man im Hauptmenü einfach den Pfadnamen der zugehörigen Objektdatei, also etwa
GENAU.O
eingibt. Diese Datei ist eine Systemdatei im Sinne von ProDOS, so daß man sogar „SYSTEM.PASCAL" (jetzt HELLO.SYSTEM) auf der Diskette durch ein eigenes Programm auf der Diskette ersetzen könnte, das dann beim Bootvorgang automatisch aufgerufen werden würde. Unverzichtbar für ein erfolgreiches Starten des Programms ist auf jeden Fall das Vorhandensein der zusätzlich zum Objektcode benötigten Runtime-Library **LIB** in demselben Volume-Directory (Version 1.0) bzw. in demselben Subdirectory (Version 1.1).

3. Compiler und Objektcodedatei

Beim Compilieren benötigt der Compiler vermutlich drei Läufe, was die hierbei erzeugbaren Assemblerausdrucke und ein Dump des Compilers nahelegen. Im Tutorial Manual gibt es hierüber keine Hinweise; ein Reference Manual liegt mir nicht vor. Zunächst scheint der Quelltext in einen Zwischencode, der dem P-Code des UCSD-Systems sehr ähnelt, übersetzt zu werden. Dies bedeutet offenbar auch, daß durch das Kyan-System auf dem 6502-Prozessor eine Stackmaschine ähnlich der P-Maschine simuliert wird. In einem weiteren Lauf wird der Zwischencode vermutlich in einen Assembler-Quelltext umgesetzt. Auf jeden Fall gibt es zu jedem Befehl des Zwischencodes im Compiler ein entsprechendes kleines Assemblerprogramm. In einem dritten Lauf wird dann endlich die Objektcodedatei erzeugt. Dieses Vorgehen scheint sehr umständlich zu sein und ist sicherlich für die im Vergleich zu Turbo-Pascal langen Compilierungszeiten mit verantwortlich. Allerdings hat Kyan Software so eine äußerst elegante Möglichkeit verwirklicht, Maschinenprogrammteile in ein Pascal-Programm einzubinden.

3.1. Assembler und Compiler

Für die folgenden Betrachtungen soll das Programm **P3** zugrundegelegt werden, das die 80-Zeichenkarte des Apple einschaltet. Die hier eingefügte Prozedur „Procedure pr (slot: integer);" stammt von der Kyan-Diskette. Anhand des Programmes P3 sieht man auch, daß Kyan-Pascal die Schnittstellen zu Peripheriegeräten so anspricht, wie es Applesoft oder der Apple-Monitor tun. Wählt man beim Compilieren die Option „-L1", hier also
P3-L1
(V1.1: P3-L-S1),
so erzeugt der Compiler die Ausgabe einer Assemblerdatei auf dem Drucker, falls dieser wie üblich in Steckplatz 1 angeschlossen ist. Ein Abspeichern dieser Datei auf Diskette ist aus den o.g. Gründen leider noch nicht möglich.
Eine Untersuchung der Assemblerdatei ist trotzdem sehr lehrreich und macht das Kyan-Pascal-System sicherlich für diejenigen interessant, die Informatik unterrichten und hierbei auch auf die Arbeitsweise und den Aufbau von Compilern eingehen wollen.
Betrachten wir nun also die Datei „P3": Zunächst ist hier am Quelltext zu erkennen, wie Kyan-Pascal Maschinenbefehle in ein Pascal-Programm einbindet. Die Folge der Maschinenbefehle ist mit **#A** zu eröffnen und mit # abzuschließen. Diese Steuerzeichen müssen in der äußersten linken Spalte einer logischen Zeile im Quelltext stehen und steuern den Compiler. Sehr angenehm ist, daß das Maschinenprogramm als Folge von Assembler-Mnemonics einzugeben ist und nicht, wie bei den Inline-Anweisungen von Turbo-Pascal, schon als fertig assemblierter Hexcode. Gerade durch diese Eigenschaft könnte Kyan-Pascal zu einem Leckerbissen für Freunde der Maschinensprache werden. Das Tutorial Manual gibt an, daß man Marken in Maschinensprachteilen nicht mit dem Buchstaben L beginnen lassen soll, weil man hierdurch den Compiler verwirrt, der bis auf einige Ausnahmen sämtliche Marken mit L beginnen läßt und dann durchnumeriert (L1, L2 usw.). Weiterhin muß in Maschinenprogrammteilen das X-Register des Prozessors gerettet werden, da es als Stapelzeiger in der Null-

seite bei der Übergabe von Daten und Parametern bzw. Adressen von Variablen benutzt wird. Anders als im P-System wird hierfür nicht der CPU-Stapel verwendet.

Wenn im Hauptmenü
P3.O
eingegeben wird, so wird die Objektdatei P3.O, also das 6502-Maschinenprogramm, ab $2000 in den Speicher geladen und gestartet. Das Programm überprüft zunächst, ob sich in dem durch das ProDOS-Präfix angegebenen Directory die Datei LIB, also die Laufzeitbibliothek des Pascal-Systems, befindet. Wird LIB gefunden, so wird versucht, sie in zwei Teilen in den Hauptspeicher zu bringen, von $0800 bis $1FFF einschließlich und von $B000 bis $BAFF einschließlich. Ab $BF00 beginnt ProDOS. Mißlingt dies oder ist LIB nicht vorhanden, so verzweigt das Programm zur Marke ERROR und von dort aus in den Apple-Monitor. Dies wird durch den Code von $2000 bis $207F bewirkt. Unter Version 1.1 ist für diesen Vorgang der Code von $2000 bis $20AC verantwortlich, wobei lediglich LIB woanders gesucht wird.

Läuft soweit alles erfolgreich, verzweigt das Programm in die Systemroutine ab $0800, die Startroutine. Diese nimmt zunächst die Rücksprungadresse vom CPU-Stapel, prüft anhand der ab $2082 (Version 1.1: $20AF) abgelegten Parameter, ob das eigentliche, in Maschinencode umgesetzte Pascal-Programm zu einer anderen Startadresse zu verschieben ist, und führt letzteres gegebenenfalls auch durch. Vor einer eventuellen Verschiebung befindet es sich ab $208F (Version 1.1: $20B3) im Speicher. Anschließend legt die Startroutine eine neue Rücksprungadresse auf den Stack und führt einen Sprung zum Beginn des eigentlichen Hauptprogramms durch, das mit RTS endet und sich so als Fortsetzung der Startroutine herausstellt. Nach dem Rücksprung aus dem Pascal-Hauptprogramm werden in einer anderen Systemroutine, der Schlußroutine, alle noch offenen Dateien geschlossen und in das Hauptmenü zurückgesprungen.

Der Kyan-Compiler setzt die von $2000 bis $208E (Version 1.1: $20B2) stehenden Befehle vor jedes compilierte Programm und bindet es so zwischen Start- und Schlußroutine ein, wodurch ein ordnungsgemäßes Funktionieren des Systems gewährleistet wird. Dies bedeutet aber auch, daß bei jedem Start eines compilierten Pascal-Programmes die gesamte Systembibliothek neu von der Diskette geladen wird, was merklich Zeit kostet. Hier empfiehlt es sich, eine RAM-Disk einzusetzen.

Sehr schön an den vom Compiler erzeugten Assemblerdateien finde ich, daß jeweils als Kommentar die aktuelle Zeile des Pascal-Quelltextes auftritt, gefolgt von ihrer Umsetzung in Maschinencode. Gerade diese Eigenschaft dürfte den Kyan-Pascal-Compiler für jeglichen Unterricht interessant machen, der sich in irgendeiner Weise mit Compilerbau befaßt. Dies dürfte auch die Arbeit sehr erleichtern, wenn man den vom Compiler erzeugten Code optimieren möchte, auch wenn hierzu eine Dokumentation der Laufzeitroutinen in LIB wünschenswert wäre. Vielleicht kann diese Eigenschaft auch über einige in den Compilern der Versionen 1.0 und 1.1 noch vorhandene Kinderkrankheiten hinwegtrösten.

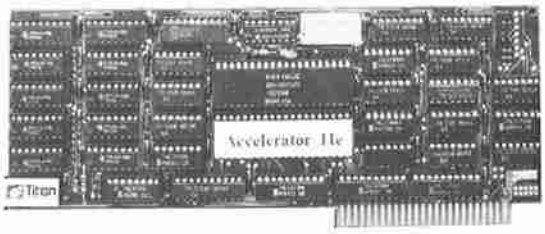
3.2. Labels

Der Compiler definiert einige wichtige Marken, auf deren Bedeutung ich hier kurz eingehen möchte. Es sind dies neben PATHNAME und ERROR die Marken GLOBAL, LOCAL, SP, P, T, MLI, L0, L1, L2, L3 und L4.
– **GLOBAL** ($B000) kennzeichnet den ersten dem Benutzerprogramm nicht mehr verfügbaren Speicherplatz und dient als Referenzmarke für die sämtlichen Variablen zuzuordnenden Speicherplätze. So dient GLOBAL-1 ($AFFF) als Pufferspeicher für die Datei INPUT (Tastatur), GLOBAL-2 ($AFFE) als Pufferspeicher für die Datei OUTPUT (Bildschirm), von GLOBAL-3 an abwärts werden die im Programm auftretenden Variablen angelegt, und zwar bis GLOBAL-L4 die globalen Variablen, dann dynamisch die bei Prozeduraufrufen erforderlichen lokalen.
– **LOCAL** ($0002) und **SP** ($0004) sind Zeiger auf den Variablenstapel, den sie beim Aufruf von Prozeduren mittels einer linearen, verketteten Liste verwalten, wodurch unter anderem auch Rekursion organisiert wird. Mit Hilfe dieser Zeiger erfolgt außerdem der Zugriff auf lokale und aktuelle Parameter von Prozeduren, denn diesen Parametern werden, anders als den globalen, keine festen Speicherplätze zugewiesen.
– **P** ($0080) ist die Startadresse für die bei ProDOS-Aufrufen erforderlichen Parameter während des Programmstarts. Ob P sonst noch vom System benötigt wird, ist mir noch nicht klar.
– **T** ($0010) dient als Zwischenspeicher insbesondere bei der Aktivierung von Prozeduren und darf offenbar auch von Maschinenprogrammteilen durch den Anwender benutzt werden.
– **MLI** ($BF00) ist die Eintrittsadresse für ProDOS-Systemaufrufe.
– **L0** gibt die Speicherstelle an, von der ab unmittelbar nach dem Einlesen der Objektcodedatei der Maschinencode des eigentlichen Pascal-Programms steht.
– **L1** bezeichnet die Speicherstelle, an der der Code nach einer eventuellen Verschiebung beginnen sollte, sonst stimmt L1 mit L0 überein.
– **L2** gibt den niedrigsten vom Programmcode nicht mehr belegten Speicherplatz an. Hier beginnt also die Halde für dynamische Variablen.
– **L3** bezeichnet die Startadresse des Pascal-Hauptprogrammes. Falls im Programm Funktionen oder Prozeduren vereinbart oder durch die Include-Option des Compilers (#I) eingebunden werden, sind L1 und L3 verschieden voneinander.
– **L4** gibt die Anzahl der von globalen Variablen des Pascal-Programmes belegten Bytes an, wobei vom System selbst definierte globale Variablen wie INPUT und OUTPUT sowie nicht näher dokumentierte globale Hilfsvariablen mitgezählt werden.
Während der Compilierung erzeugt der Compiler weitere Marken je nach Bedarf, die alle die Form Ln haben, wobei n die Marken durchnumeriert. Hierdurch ist das Verbot für den Anwender begründet, eigene Marken in Maschinenprogrammteilen mit L beginnen zu lassen.

4. Der Sprachumfang

Wie UCSD- und Turbo-Pascal kennt Kyan-Pascal keine Funktionen und Prozeduren als formale Parameter bei Funktionen und Prozeduren.
Alle im Standard-Pascal verlangten Wortsymbole sowie die Compilerdirektive FORWARD sind vorhanden. Bei den Spezialsymbolen müssen für die geschweiften Klammern die Ersatzdarstellungen „(*" und „*)" benutzt werden, ab Version 1.1 sind auch die geschweiften Klammern zulässig. PACK und UNPACK fehlen wie in UCSD-Pascal und Turbo-Pascal. Ob in jedem Fall automatisch gepackt wird, ist nicht klar. Das Tutorial Manual schweigt sich hierüber aus. Daten vom Typ ARRAY ... OF CHAR werden jedenfalls gepackt dargestellt.
Weiterhin fehlt wie in Turbo-Pascal die Standardprozedur PAGE, obwohl das Handbuch auf sie verweist; sie soll später einmal eingebaut werden.
Soweit ich feststellen konnte, arbeiten die Funktionen und Prozeduren zur Dateiverwaltung ordnungsgemäß. Die nicht zum Standard-Pascal gehörende Prozedur CLOSE ist nicht realisiert; sie ist hier auch unnötig, da bei jedem Aufruf von RESET oder REWRITE eventuell betroffene offene Dateien automatisch geschlossen werden. Löschen von Dateien ist so nicht möglich. Als Erweiterung ist die Prozedur SEEK eingebaut.
Mittels der Erweiterung CHAIN läßt sich an ein Pascal-Programm ein weiteres Pascal-Programm oder eine andere ProDOS-Sy-

Accelerator™ IIe macht Ihren Apple® II, II Plus oder IIe dreieinhalbmal schneller.



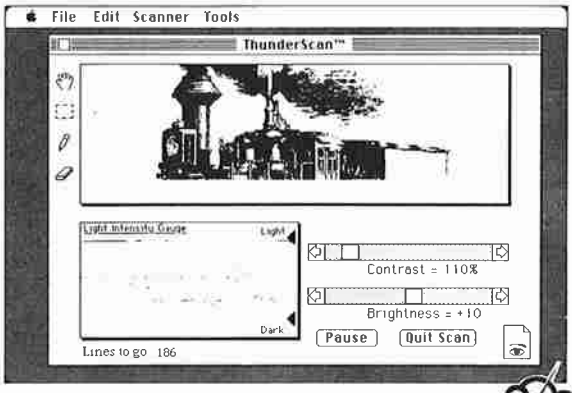
Jetzt laufen VisiCalc®, Apple Writer, PASCAL, BASIC, Datenbanken usw. endlich ohne langen Zeitverlust. Stecken Sie einfach die ACCELERATOR IIe Karte in irgendeinen Slot und beobachten Sie, wir Ihr Apple loslegt!
ACCELERATOR IIe besitzt seinen eigenen schnellen 6502 Prozessor und 80 K-Byte Hochgeschwindigkeitsspeicher, einschließlich einer eingebauten schnellen Sprachkarte und schnellem RAM-Speicherplatz für die ROM-Sprache.
Direkt von Pandasoft (Titan Distributor für Deutschland) oder bei Ihrem Applehändler.

pandasoft Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo–Fr 10–18 Uhr, Sa 10–13 Uhr
Telefon: 0 30/31 04 23 · Telex: 1 85 859

ThunderScan.™

Ein neues optisches Lesegerät, das beliebige Vorlagen in MacPaint überträgt: Fotos, Zeichnungen, Landkarten und Illustrationen werden in den Apple-Imagewriter eingespannt und von einem Lesekopf, der das Farbband ersetzt, abgetastet.



- 32 Graustufen
- 80 Punkte/cm Auflösung
- Übertragungsmaßstab 25 % – 400 %
- Vorlagen bis 20 x 25 cm
- Nachträgliche Veränderung des Kontrasts und der Helligkeit.

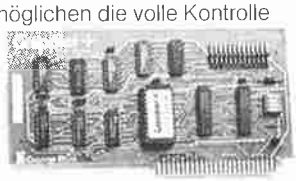
ThunderScan

pandasoft Dr.-Ing. Eden

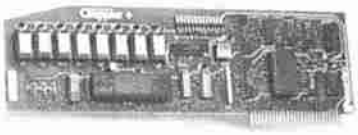
Uhlandstraße 195 · 1000 Berlin 12 · Mo–Fr 10–18 Uhr, Sa 10–13 Uhr
Telefon: 0 30/31 04 23 · Telex: 1 85 859

Orange™ Printer Interface **Druckerinterfaces** für Apple II+/e/c/III Interfaces auf dem **neuesten Stand der Technik. Kompatibel** mit allen gängigen Druckern wie: APPLE, EPSON, STAR, NEC, OKIDATA usw. Passende Treibersoftware wird über Dip-Switch ausgewählt.

Grappler + Printer Interface **Grafikfähiges Druckerinterface** das keine Wünsche mehr offen läßt. Über **2 Dutzend Kommandos** ermöglichen die volle Kontrolle über alle Möglichkeiten Ihres Druckers. Jetzt auch mit **IIe Features: Double Hires Grafics** und **80 Zeichen Dump** mittels Druckerpuffer nachrüstbar über Bufferboard.



BUFFERED Grappler + Besitzt alle Vorzüge des Grappler +, hat aber zusätzlich einen integrierten **16 K Druckpuffer,** der auf **32 oder 64 K aufrüstbar** ist.



SERIAL Grappler Printer Interface **Serielles Druckerinterface** speziell für den **Apple Imagewriter.**

HOTLINK Seriell-nach-Parallel-Wandler für den IIc im Kabel integriert.

GRAPPLER C wie Hotlink, jedoch zusätzlich Imagewriter Emulation und Grafik Software-Diskette.

pandasoft Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo–Fr 10–18 Uhr, Sa 10–13 Uhr
Telefon: 0 30/31 04 23 · Telex: 1 85 859

Sie haben einen Apple...

wir haben die Software...

und die Hardware...

wir haben die Bücher...

und die Zeitschriften*...

***Fordern Sie unseren Gratiskatalog an!**

ALLES FÜR DEN APPLE II+, IIe, IIc UND MACINTOSH

pandasoft Dr.-Ing. Eden

UHLANDSTR. 195 · D-1000 BERLIN 12
TEL.:(030) 310 423 · TELEX: 18 58 59

Autorisierter apple Fachhändler MICROSOFT Distributor

Ich besitze einen Apple. Bitte schicken Sie mir Ihren kostenlosen Katalog.
Name: .....
Adresse: .....

stemdatei anketten, wobei sich Kyan-Pascal-Programme Variablen nicht nur über Dateien übergeben können, sondern auch wie in Turbo-Pascal dadurch, daß die gemeinsamen Variablen zu Beginn aller beteiligten Progamme in gleicher Reihenfolge und mit gleichen Datentypen lückenlos vereinbart werden; andere Bezeichner dürfen vorkommen.
Die Verwaltung der Halde (Heap) erfolgt durch die Standardprozeduren NEW und DISPOSE, die auf den ersten Blick auch ordnungsgemäß arbeiten und auf der Halde eine lineare, verkettete Liste anlegen. NEW weist dynamischen Variablen Speicherplatz auf der Halde zu, der von DISPOSE dem System auch korrekt zurückgegeben wird, jedoch wurden bei meinen Tests in beiden zur Zeit vorliegenden Versionen des Kyan-Pascal für jede dynamische Variable unabhängig vom tatsächlichen Bedarf 435 Bytes reserviert, weshalb die Haldenverwaltung ernsthaften Ansprüchen bislang nicht genügt. Da sich mit der Erweiterung ASSIGN (Zeiger:POINTER; Stelle:INTEGER) Zeiger auf vom Programmierer definierte Speicherplätze richten lassen, kann man sich hier mit einer eigenen Verwaltung der Halde behelfen, aber dies ist nicht Sinn der Sache.
Bei Verbunden (RECORDs) darf im Gegensatz zu Standard-Pascal der Bezeichner im Variantenteil nicht fehlen. Ein Konstrukt der Form „... CASE BOOLEAN OF TRUE: (Feldliste); FALSE: (Feldliste) ..." ist also nicht möglich. Weiterhin werden boolesche und selbstdefinierte skalare Selektoren nicht akzeptiert, sondern nach meinen Tests nur solche vom Typ INTEGER oder CHAR. Eine wesentliche Einschränkung ist hierdurch allerdings nicht gegeben. Schwerer wiegt da schon, daß weder unter Version 1.0 noch 1.1 die WITH-Anweisung funktioniert; der Compiler stürzt hier in den Apple-Monitor ab.
Der Datentyp STRING ist anders als in Turbo-Pascal und UCSD-Pascal nicht eingebaut, sondern muß vom Programmierer wie im Standard-Pascal beispielsweise als
CONST MaxString = 40;
TYPE String = ARRAY [1..MaxString] OF CHAR;
definiert werden, jedoch arbeiten READ, READLN, WRITE, WRITELN klaglos mit irgendwelchen Zeichenketten, ohne eine Laufanweisung zu verlangen.
In einigen Textdateien der System-Diskette stehen für die Verarbeitung des so definierten Datentyps String die Prozeduren bzw. Funktionen LENGTH zur Bestimmung der Länge eines Strings, CONCAT zur Verkettung von Strings, INDEX zum Auffinden der Position eines Strings innerhalb eines anderen Strings (entspricht der POS-Funktion des UCSD-Pascal) und SUBSTRING, um Teilstrings aus einem String herauszuziehen (entspricht der UCSD-Prozedur COPY) zur Verfügung.
Mit Hilfe einer weiteren Textdatei, die beispielsweise durch die Include-Anweisung des Compilers „#I HIRES.I#" in ein Programm eingebunden werden kann, sind für die hochauflösende Grafik die Prozeduren HGR zum Einschalten und Löschen der Grafikseite 1, TX zum Einschalten der Textseite, PLOT zum Zeichnen eines Punktes auf der Grafikseite und DRAW zum Zeichnen von Strecken auf der Grafikseite verfügbar.

5. Laufzeiten

Bei der Bestimmung der Laufzeiten wurden Kyan-Pascal 1.1, UCSD-Pascal 1.2 und Turbo-Pascal 1.0 mit einer normalen Z80A-Karte benutzt. Die Ergebnisse finden Sie in **Tabelle 1**. Weiterhin habe ich einige Untersuchungen mit der Ackermannfunktion (siehe Programm **ACKERMANN**) durchgeführt, deren Ergebnisse sich in **Tabelle 2** finden.
Bei der Wertung sollte man jedoch noch die jeweils erreichbare Rechengenauigkeit berücksichtigen.

6. Fazit

Ich selbst programmiere bislang nahezu ausschließlich in 6502-Assembler und in Pascal. Deshalb bin ich sehr froh, endlich ein Pascal-System für den Apple II verfügbar zu haben, das Maschinencode für den eigentlichen Prozessor des Apple erzeugt und außerdem recht preiswert ist. Bestechend elegant ist außerdem die Einbindung von Maschinencode in ein Pascal-Programm, die für die recht langen Compilierungszeiten entschädigen mag. Bislang entdeckte Fehler des Systems betreffen raffiniertere Programmiertechniken mit Zeigern und Verbundtypen und lassen sich mit Mitteln des Systems umgehen, auf keinen Fall mindern sie den Wert des Systems für Pascal-Neulinge.

Literatur

Pascal, Einführung in die Sprache, Norm-Entwurf DIN 66256 Erläuterungen von Klaus Däßler, M. Sommer, Springer-Verlag, Berlin, Heidelberg, 1983
Turbo-Pascal von Rudolf Herschel, Oldenbourg Verlag, München, 1985
Apple Pascal, Sprache, te-wi-Verlag, 1985
Apple Pascal, Betriebssystem, te-wi-Verlag, 1985
Apple Pascal, Addendum Pascal 1.2., te-wi-Verlag, 1985
Turbo Pascal Language Manual, Borland International, 1984
Kyan Pascal, A Programming Language, Tutorial Manual, 1985

GENAU

```
Program Genau;
{Test auf Rundungsungenauigkeiten}
var x, y : real;
    i    : integer;
begin
x := 1Ø;
y := Ø.Ø1;
writeln (chr(7));
while x <> Ø do
  begin
  x := x - y;
  write (x:2Ø:1Ø);
  if x < -Ø.1 then
    repeat write (chr(7)) until false
  end;
repeat write (chr(7));
  for i:=1 to 1ØØØ do {Pause}
until false
end.
```

ACKERMANN

```
Program Ackermann;
var wert, m, n : integer;

function Acker (m, n : integer): integer;
  var a : integer;
  begin
  if (m<Ø) or (n<Ø) then
    begin writeln (chr(7));
    writeln ('Fehler'); readln
    end;
  {write ('(',m,',',n,') ');}
  if m=Ø then a := n+1
    else
      begin
      if n=Ø then a := acker (m-1,1)
      else a := acker (m-1,acker (m,n-1));
      end;
  acker := a
  end {acker};
```

```
begin
repeat
  writeln ('Es wird die Ackermannfunktion berechnet');
  writeln ('Geben Sie m und n ein');
  readln (m,n);
  write (chr(7));
  Wert := Acker (m,n);
  write (chr(7));
  writeln ('A(',m,',',n,') = ',wert)
until false
end.
```

P3

```
Program P3;

{8Ø-Zeichenkarte einschalten}
{Diese Prozedur simuliert PR#s}

procedure pr (slot: integer);
begin
#A {Anfang 65Ø2-Assembler}
 STX T
 LDY #3
 LDA (SP),Y
 JSR $FE95   ;OUTPORT
 LDX T
# {Ende 65Ø2-Assembler}
end;

begin
 pr (3);
 write ('Hier muss sie sein.');
 readln
end.
```

Tabelle 1: Laufzeiten

Operation	Zeiten in ms		
	Kyan-Pascal	UCSD-Pascal	Turbo-Pascal
Berechnung der Wurzel einer ganzen Zahl	11Ø..129	39..44	21..34
Division c := a/b	11.2	4.25	5.Ø4
Multiplikation ganzer Zahlen	Ø.79Ø	1.Ø	Ø.314..Ø.544
Addition ganzer Zahlen	Ø.194	Ø.344	Ø.Ø44
Mult. reeller Zahlen	4.3..4.9	2.4..2.8	etwa 3.Ø
Add. reeller Zahlen	1.39..1.86	1.35..1.48	Ø.51..Ø.66
Leere FOR-Schleife, ein Durchgang	Ø.21	Ø.7Ø	Ø.Ø6

Tabelle 2: Ackermannfunktion:

	Kyan-Pascal	UCSD-Pascal	Turbo-Pascal
Ackermann(3,2)	Ø.8 s	1.8 s	Ø.6 s
Ackermann(2,22)	1.4 s	3.4 s	1.1 s
Ackermann(2,23)	in 1,5 s berechnet, Absturz	3.6 s	1.3 s
Ackermann(3,4)	hängt	29.2 s	8.6 s
Ackermann(3,5)	hängt	Fehler	berechnet

Tabelle 3: Wichtige Entstehungsdaten

ED	Version 1.Ø	23.Ø4.1985	(4Ø-Z/Z-Editor)
PC	Version 1.Ø	14.Ø4.1985	(Pascal-Compiler)
LIB	Version 1.Ø	Ø7.Ø4.1985	(Library)
ED	Version 1.1	16.Ø6.1985	
E8Ø	Version 1.1	26.Ø6.1985	(8Ø-Z/Z-Editor)
PC	Version 1.1	19.Ø6.1985	
LIB	Version 1.1	<NO DATE>	
ED	Version 1.2	Ø6.Ø7.1985	
E8Ø	Version 1.2	Ø6.Ø7.1985	
PC	Version 1.2	27.Ø7.1985	
LIB	Version 1.2	26.Ø6.1985	

Preiswerte Begleitdisketten

Bd. 1: DM 28,–; Bd. 2: DM 28,–

DM 28,–

DM 28,– (Neue Diskette für 3. Aufl.)

Hüthig Software Service
Postfach 10 28 69 · 6900 Heidelberg

aaa electronic gmbh

Neue Preise #1

IMB-PC/XT-COMPATIBLE COMPUTER + ZUBEHÖR
Mainboard XT f. 16bit + 256K-RAM + 8 Slots, Leerkarte 79,—
Mainboard XT-640 K (on Bord) f. 16 Bit + 8 Stots Leerkarte 89,—
Disk Controller leer (f. 2x360 KB-Drives) 29,—
Disk Controller leer (f. 4 x 360KB-Drives) 34,—
Monochrome Graph. + par. Printer Leerkarte (Hercules comp.) 55,—
Color RGB + Video Graphik Leerkarte 55,—
Parallel Printer Leerkarte 29,—
512K-RAM Leerkarte 55,—
Multifunktions Leerkarte (Printer, Ser, Game Uhr gepuffert) 49,—
Disk-Multi I/O Leerkarte, (f. 3x360 K-Drive, 2xRS 232 1xpar. Printer Int., Game/Joystick Port, Real Time Clock/Kalender gepuff.) 59,—
384 K-Multifunktions Leerkarte, (1xpar. Printer Int., Game/Joystiek Port, Real/time Clock/Kalender gepuffert) 55,—
Prototype/Lochraster Leerkarte, durchkontaktiert 88,—
Mainboard XT-256 f. 16 bit CPU, OK (bis 256K-an board, oder 640 K mit Addr. auf Karte aufr.) + Boot-ROM + 8 Slots best. + geprüft 359,—
Mainboard XT-256 w.o. jedoch 256 K – vollbest. und geprüft 439,—
Mainboard XT-640 f. 16 bit CPU, OK (bis 640 K-on board aufr.) 389,—
Mainboard XT-640 w.o. jedoch 256 K-RAM best. + gepr. 469,—
Mainborard XT-640 w.o. jedoch 640 K – vollbest. und geprüft 749,—
64KB-RAM-Chip-Aufrüstsatz (9 Stück) einzeln geprüft 21,—
256 KB-RAM-Chip-Aufrüstsatz (9 Stück) einzeln geprüft 159,—
Disk-Controller gepr. (f. 2x360 KB Drives) (Extra Kabel = 39,—) 99,—
Disk Controller gepr. (f. 4 x 360 KB-Drives)(Extra Kabel = 39,—) 109,—
Monochrome Karte geprüft 235,—
Monochrome Graph. + par. Printer Karte gepr. (Hercules comp.) 315,—
Color RGB + Video Graphikkarte gepr. 175,—
Parallel Printer-Karte gepr. o. Kabel (Extra Kabel = 49,–) 77,—
512K-RAM-Karte gepr. (0KB) 135,—
384K-Multifunktionskarte, RS 232, par. Printer, Interf. Game/Joystick Port, Real Time Clock gepuff., 0K + RAM-Disk/Spooler/Uhr Softw. 325,—
Multifunktionskarte (Par. + Ser. + Uhr gepuffert) gepr. 289,—
Diek Multi I/O Karte (f. 2x360 K-Drive, 2xRS 232, 1xpar. Printer Int., Game/Joystick Port, Real Time Clock/Kalender gepuff.) 397,—
Profi-Funktionstastatur dt. oder ASCII + 15er Block ab 215,—
135W/150W-Netzteil mit eingebautem Ventilator, Kurzschluß. 219,—/279,—
360KB-Diskdrive Slimline (Track Access Time <2 m S) 319,—
PC/XT – KOMPLETT-SYSTEME EINGEBAUT UND GEPRÜFT
XT-256, 16 bit-Rechner (inel. Boot-Eprom) 128K-Ram im
IBM-look-like Met.-Gehäuse+Profi-Fkt. Tast.+135 W-Netzt. gepr. 999,—
XT-256, w.o. + Contr. +360KBDrive+Color RGB+Vid. Graphic-Karte 1627,—
XT-256, w.o. jedoch statt Color mit Monochrome Karte 1687,—
XT-256, w.o. jedoch mit Monochrome-Graphic-Printer-Karte 1767,—
PC 10-Commodore 256 K+2 Laufwerke + Monitor + dt. Tast. 4259,—
640 K-Speichererweiterungs-Set für PC 10 369,—

APPLE-BUS COMPUTERPLATINEN + PERIPHERIE
Motherboard II 48K leer, durchkont. mit Lötstopplack 57,—
Motherboard II w.o. jedoch **64 K** leer 78,—
M-board II 48K, gesockelt, vollbest. + gepr. 337,—
Motherboard II 64K gesockelt, vollbest.+gepr. 375,—
M-board II 64K + Z80 CPU, gesockelt, v. best. + gepr. 378,—
M-board Ile 64K vollbestückt + gepr. 435,—
APOLLO II **Kunststoff-Leergehäuse** 85,—
APOLLO II **Gehäuse** incl. Funkt.-Tast. 217,—
Superstarkes Netzteil – Kurzschlußfest 115,—
Leerplatine: 16K-RAM, Z80/CPU-, Controller DOS 3.3, Parallel-Druckerkarte, Ser. Int. V24, 80Z/24Z-Karte, PAL-Karte je 18,95
Entwicklungslochraster-Leerplatine durchkontaktiert 27,—
16K-Speicher-Karte geprüft 55,—
APPLE orig. 16K-Language Karte gepr. 125,—
Z-80 CPU-Karte geprüft 64,—
Controller DOS 3.3-Karte geprüft 63,—
Auto-Controller DOS 3.2/3.3-Karte gepr. 119,—
Erphi AFDC 2 Controller + Autopatch: Softw. + Handbuch 284,—
Parallel-Drucker-Karte geprüft 74,—
Par.-Drucker-Grappler comp. Leerplatine 27,—
Par.-Drucker Karte-Grappler comp. geprüft 169,—
8bit par. Grappler-Int. + 64K-Buffer + Kabel. gepr. 445,—
Seriell-Interface-Karte V24 geprüft 88,—
Super-Seriell Leerkarte 34,—
Super-Seriell-Interface Karte gepr. 169,—
80-Zeichen/24-Zeilen-Karte geprüft 119,—
80Z/24Z-Karte + Softswitch-Schalter Leerplatine 24,—
80Z/24Z-Karte + Softswitch-Schalter, gest. Scharf geprüft 127,—
80Z/24Z-Softwitch Karte gepr. für Ile 49,—
80Z/24Z+64K-Ram+Softw. Leerkarte f. Ile 27,—
80Z/24Z+64K-Ram+Softsw.-Karte best.+gepr. f. Ile 69,—
IEEE-488 Int. Karte gepr. 345,—
PAL-Modulator/Color-Karte geprüft 99,—
RGB-Color Karte gepr. 129,—
128K-Speicher-Leerkarte 29,—
128K-Speicher-Leerkarte + Software + Manual 99,—
128K-Speicher-Karte + Software geprüft+Manual 264,—
256K-Speicher Leerkarte 34,—
256K-Speicher Leerkarte+Software+Manual 109,—
256K-Speicher-Karte + Software geprüft+Manual 455,—
Maus incl. Software 179,—
EPROM-Burner 2716/32/64) geprüft 109,—
6522-VIA Karte geprüft 135,—
Clock Karte geprüft 94,—
Speech Karte geprüft 53,—
Wild Karte geprüft 59,—
AD/DA-8Bit Karte gepr. 375,—
Modem Euronorm-Karte gepr. + Software + Man. 397,—
UHF/TV-Modulator universell 27,50
Lüfter anclipsbar (220 V) 79,—
Dataphon s21d Akustikkoppler-FTZ mit RS 232/V24 Anschl. 275,—

APOLLO//(48K) + UHF-Mod. + Gr/Kl vollbest. + gepr. + F-T. 617,—
APOLLO//(48K) + Disk II F + Contr. + 12"-Monitor 1.295,—
APOLLO//A (48K) + Gr/Kl + 1Ser-Tastatur vollbest. + gepr. + F-T. 677,—
APOLLO//A (48K) + Disk II F + Contr. +12"-Mon. 1.355,—
APOLLO// ASKF (48K) vollbest. + gepr. Sep. Keyboard + Gr/Kl + 1Ser-Block mit Funkt.-Tasten im „IBM-look like" Gehäuse 835,—
APOLLO// ASKF (48K) w.o. + Disk II F + Contr. 1.235,—
Apollo II ASFK (48K) w.o. + 2xDisk II F + Contr.+12" Mon. 1.855,—
Aufpreis f.o. Systeme von 48K zu 64K 59,—
Aufpreis von 48K zu 64K+Z80 CPU 89,—
Apollo Ile (64K) PAL vollbest. und geprüft 725,—
Apollo Ile (128K)+80Z/24Z-Karte geprüft 825,—
Apollo Ile ASKF ab 876,—
Apollo Ile (64K) + Disk II F + 12" Monitor 1.439,—
APPLE Ile 128K+80Z/24Z Karte+Softw. 1.998,—
Epson-Duodisk II F-Slimline 2x143 KB mit Geh. + Kabel 799,—
Disk//+Contr. + Kabel DOS 3.3 (Siemens) im Geh. 539,—
Disk//(Siemens) im Geh. + Kabel 478,—
Disk//F + Contr. + Kab. DOS 3.3 (1/2 Höhe) im Geh. 379,—
Disk//F (1/2 Höhe) i. Geh. + Kabel (Trak Access <2 m S) 318,—
Erphi DuoDisk 1,2 Mbyte incl. Netzteil + Erphi APPC 2-Contr. 1.789,—

APPLESOFT + Turorial + Reference Manual engl. 75,—
APPLE-DOS 3.3 Manual engl. 25,—
APPLE-DOS 3.3 Hb. dt. 45,—
APPLE-Pascal Reference + Operating Manual engl. 48,—
APPLE-Pascal Reference Hb. dt. 49,—
APPLE-Pascal Language Hb. dt. 39,—
APPLE-Fortran Manual engl. 69,—
CP/M-Softcard Vol. I + II, Manual engl. 69,—
APPLE II+Ile-Anwenderhandbuch deutsch (te-wi) 66,—
Z80 Karte gepr. f. APPLE//c + RAM-Disk-Software + Hb. 489,—
Macintosh System (128K) 5.477,—
Macintosh System (512K) 6.095,—
Macintosh Umrüstung von 128K auf 512K 678,—
SANYO MSX-Computer MPL 64 dt. Tastatur 685,—
Epson LX 80, 8 Bit/par. (Extra Traktor = 69,–) 859,—
EPSON FX 85F/T+, 8bit/parallel 160 Z/S. NLQ + IBM Komp. 1.285,—
EPSON FX 105+, 8bit/parallel 15" sonst w.o. 1.715,—
EPSON Traktoraufsatz für FX 80 + FY 85 129,50
APPLE/EPSON-Drucker Graphic-Interface + Kabel 169,—
MX 80/82, FX/RX 80 Spezialfarbband-Kassette 15,95
BROTHER-Typenraddr. HR 15XL 8bit/par. neuestes Mod. 1.387,—
STAR SG10, NLQ- + IBM-Zeichensatz, 8bit/par. 998,—

Wir führen verschiedene Monitore von Zenith, Phillips und Sanyo mit Video – TTL (IBM Komp.) oder Coloreingang.

Disketten in Box + Aufkleber i. Wahl, 10er Pack/100er Pack./ Stückpreis:

5 ¼"	Verstärkt	BASF	3M SCOTCH	FUJI	DATA MAGNETICS	NEUTRAL
1X, SS/SD		3,69/ 3,49			2,86/2,76	
1D, SS/DD		3,89/ 3,69	4,88/ 4,58	4,27/ 3,97	3,96/3,76	2,75/2,65
2D, DS/DD		5,69/ 5,49	6,38/ 5,88	5,87/ 5,47	4,76/4,46	3,85/3,65
1D/96TPI		5,59/ 5,39	7,18/ 6,68			
2D/96TPI		7,09/ 6,99	7,88/ 7,48	7,87/ 7,67		
2HD/1.2MB		13,49/12,69		12,87/12,57		
1X, Hard		4,29/ 4,19				
3						
1X, SS/SD		4,49/ 4,39	6,68/ 6,48	6,97/ 6,67		
1D, SS/DD		4,79/ 4,59	8,58/ 8,28			
2D, DS/DD		7,09/ 6,89	9,88/ 9,58	10,17/ 9,77		
1X, Hard		**4,49/ 4,39**	**6,68/ 6,48**			
3						
Einseitig		10,99/ 9,99	10,08/ 9,88	7,67/ 7,47		
Zweiseitig		14,99/14,29	15,98/15,18			

Das sind Preise!

3M-Scotch 5¼" **Flüssig-Reinigungsset** (2 Disk.) zweiseitig 48,95
5¼" Disketten-Archivbox für 10 Disks (f. 3¼/0 St.) 4,75/49,50
5¼" Disk-Kartelkasten Kunststoff (ca. 80 Disk) 26,50
5¼" Disk-Kartelkasten Kunststoff (ca. 80 Disk) Rauchglas 39,95
5¼" Disk-K.K.Rauchglas, abschließb. (f. ca. 80-100 Disks) 47,50
2000 Bl. Tabpapier (24 cm x 12" einf.) weiß oder grün/weiß perf. 45,—
4000 Et.-Aufkl. doppelr. (107 x 36 mm auf 240 x 12" perf. Trägerp.) 66,50
4000 Etik.-Aufkl. einreih. (107 x 36 mm auf 125 x 12" perf. Träg.) 65,—

OSZILLOSCOPE HAMEG ab Lager

Bei Vorauszahlung frei Empfangsstation unversichert in der BRD, ausgenommen Papier und Etiketten, sonst N.N. + V.S. ab DM 30,—— Preise incl. MwSt.
Öffnungszeiten: Mo, Di, Do, Fr v. 10-18 h, Mi u. Sa v. 10-14 h, Ia Sa v. 10-18 h.
Telef. Best.: Mo, Di, Do, Fr von 10-19 h, Mi u. Sa wie Öffnungszeiten.
GEWÄHRLEISTUNG: 6 Monate, auf alle bei uns gekauften Geräte, durch unsere eigene Service-Werkstatt.
REPARATUREN an Apple + Compatible Geräte + Zubehör führt unser Spezialistenteam garantiert zuverlässig + besonders kostengünstig aus.
Sprechen sie mit uns. Kostenvorschlag auf Wunsch!

aaa electronic gmbh
Telex: 07 72 642 aaa-d
Habsburgerstraße 134
7800 FREIBURG, Tel (07 61) 27 68 64
Bauelemente – Bausatze – µP's
Meßgerate – Zubehör – Fachliteratur
Fachgeschaft fur Elektronik + Mikrocomputer

Der Apple II als Werkzeug für den Betriebswirt

Teil 1: Das Simplex-Verfahren in der Controllerpraxis

von Dipl.-Betriebswirt Willy Holtkamp

Controller sind in der Regel keine Assemblerprogrammierer. Sie bedienen sich meist einer Hochsprache, vornehmlich eines BASIC-Dialektes (z.B. Applesoft-BASIC). Der Verfasser erläutert zuerst an einem kleinen Beispiel eine Aufgabe aus der Controllerpraxis und zeigt dann Möglichkeiten auf, die dem Controller die Arbeit erleichtern.

1. Aufgaben des Controllers

Die Position des Unternehmenscontrollers hat in den letzten Jahren eine beachtliche Aufwertung erfahren. Bei der Controllerfunktion handelt es sich „um einen Lotsen- oder Navigationsdienst zum Ziel Gewinn. Der Controller ist also ein wirtschaftlicher Ratgeber des Managements. Seine Aufgabe besteht in der fachlichen Interpretation der Zahlen und betriebswirtschaftlichen Zusammenhänge sowie in der Überzeugung und Motivation für die in den Zahlen liegenden Konsequenzen." (1)

Mit dem gewachsenen Einfluß ist gleichzeitig eine Steigerung der Arbeitsbelastung verbunden. Deshalb haben sich immer mehr Controller einen Personalcomputer angeschafft, von dem sie in erster Linie Robustheit, einfache Handhabung, Ausbaufähigkeit und Vielseitigkeit in der Anwendung erwarten. Der Apple II hat seit seiner Markteinführung seinen außerordentlich hohen Nutzwert in der Controllerpraxis unter Beweis gestellt.

Das folgende kleine Beispiel, das ich aus (2) entnommen und aufbereitet habe, soll dem Außenstehenden einen Einblick in die Controllerpraxis vermitteln und den Nutzen einer Verbindung moderner Rechenverfahren und Computeranwendung für das Unternehmen verdeutlichen.

2. Ein fiktives Simplex-Beispiel

Der Controller Kraus hat von der Geschäftsleitung der Schmalhans GmbH, Düsseldorf, den Auftrag erhalten, einen Vorschlag für eine rasche **Gewinnverbesserung** auszuarbeiten.

Die Ermittlungen von Kraus ergeben, daß die Schmalhans GmbH lediglich 4 Produkte (P1-P4) auf 4 Fertigungsstellen (A-D) herstellt und verkauft. Die Situation der letzten Periode stellt sich dar wie in **Abb. 1** wiedergegeben.

Das Unternehmen erzielte somit einen **Gesamtdeckungsbeitrag** in Höhe von DM 1.354.705 in der letzten Periode. Der Deckungsbeitrag ist die Differenz zwischen Nettoumsatz und variablen Kosten und soll die Fixkosten, die Steuern und den erhofften Gewinn abdecken.

Bei der Datensichtung hat der Controller Kraus herausgefunden, daß die Marktchancen der Schmalhans GmbH beträchtlich größer sind, als der Geschäftsleitung bisher bekannt war. So halten die Verkäufer folgende **Absatzhöchstmengen** je Produkt für realisierbar:

P1 max. 3500 Stück,
P2 max. 9000 Stück,
P3 max. 11000 Stück,
P4 max. 2600 Stück.

In Verbindung mit den nicht voll ausgelasteten Fertigungsstellen A-D eröffnet sich hier ein Ausweg für die Schmalhans GmbH.

Kraus beschließt, das **deckungsbeitragsoptimale Produktmix** mittels der **Simplex-Methode** (lineare Programmierung) zu bestimmen. Ausgehend von den Daten der Vorperiode und den gerade ermittelten Absatzhöchstmengen, stellt Kraus sein **Simplex-Tableau** auf und lädt sein Optimierungsprogramm in den Apple II+. Zuerst dimensioniert er das Feld IB (9) und dann die Matrix A (9,13) und tippt dann die Daten ein. Die Matrix A (9,13) weist dann das in **Abb. 2** dargestellte Bild im Speicher auf (sinnbildlich).

Die ersten 4 Zeilen beinhalten die Koeffizienten der Fertigungsstellen A-D und geben den Zeitverbrauch je Produkt in Minuten an. Die Zeilen 5-8 berücksichtigen die Absatzhöchstgrenzen je Produkt wie oben besprochen. In der letzten Zeile sind die Deckungsbeiträge je Produkt vermerkt. Kurz nachdem der Controller Kraus „RUN" getippt hat, liegt die optimale Lösung vor. **Abb. 3** zeigt den Computer-Output, während **Abb. 4** die Lösung in Tabellenform wiedergibt.

Danach sollten von P1 insgesamt 2780 Stück, von P2 insgesamt 4885 Stück, von P3 11000 Stück und von P4 insgesamt 2315 Stück gefertigt und verkauft werden. Dies würde den Gesamtdeckungsbeitrag von DM 1.354.705 auf immerhin DM 1.469.808 steigern (= 8,5% Steigerung). Aus der Tabelle ist weiterhin abzulesen, daß die Fertigungsstellen A, B, D voll ausgelastet sind. Ein Vergleich der zu produzierenden Mengen mit den Absatzhöchstgrenzen zeigt, daß keine Absatzrestriktion verletzt wurde. Bei der Datenermittlung stellte Controller Kraus fest, daß dem speziellen Deckungsbeitrag von Produkt P1, der später mit DM 459.895 errechnet wurde, ein spezieller **Fixkostenbetrag** in Höhe von DM 506.000 gegenüberzustellen ist. Daraus resultiert ein negativer Restdeckungsbeitrag für das Produkt P1. Es ist also bald aus dem Produktionsprogramm zu nehmen. „Für die Abbaurechnung und die Abbauentscheidung kann die ‚Simplexmethode' wiederum wertvolle Informationen liefern" (2b). Hierbei gilt folgende Überlegung: Wenn die Summe der entfallenden speziellen, d.h. langfristig abbaubaren Fixkosten – hier sind das DM 506.000 – höher ist als die Summe fortfallender Deckungsbeiträge, dann würde die Abbauentscheidung das Unternehmensergebnis verbessern.

Kraus stellt nun ein neues Simplex-Schema auf und läßt dabei Produkt P1 und die Fertigungsstelle A außer acht. Das Ergebnis kann aus der **Abb. 5** abgelesen werden.

Der Gesamtdeckungsbeitrag ist von DM 1.469.808 auf 1.049.942 gesunken, also um DM 419.866. Zieht man nun von den abbaubaren speziellen Fixkosten in Höhe von DM 506.000 diese wegfallenden Dekkungsbeiträge von DM 419.866 ab, so erhält man die Summe DM 86.134, die die Ergebnisverbesserung der Schmalhans GmbH darstellt.

Dieses kleine Beispiel sollte zeigen, daß derjenige Unternehmer im Vorteil ist, der moderne Rechenverfahren und Computer benutzt. Immerhin kann die Schmalhans GmbH DM 1.469.808 - DM 1.354.705 = DM 115.103 in einem Monat mehr verdienen, ohne sich dabei groß anstrengen zu müssen. Multipliziert man diese Zahl mit 11 Arbeitsmonaten, so erhält man eine Ergebnisverbesserung von DM 1.266.133.

3. Verbesserte Simplex-Berechnungen

Im Normalfall sind die zu bearbeitenden Matrizen beträchtlich größer als soeben dargestellt. Mit zunehmender Matrixgröße steigt der Rechenaufwand und damit auch der Zeitbedarf, und zwar überproportional zur Matrixausdehnung. So sind z.B. Mehrperioden-Optimierungsmodelle mit 100 Zeilen zu je 120 Spalten keine Seltenheit. Und wenn die Frage geklärt werden muß, welche Produkte in Werk 1 und welche Produkte in Werk 2 am kostengünstigsten gefertigt werden können, dann wird die Bearbeitung noch umfangreicherer Matrizen notwendig.

Bei derartigen Aufgabenstellungen treten dann schnell die konzeptionsbedingten Nachteile des Apple II+ zutage, nämlich lediglich 64K RAM und eine durchschnittliche Rechengeschwindigkeit, gemessen an neueren Produkten der Computerindustrie.

Diese Feststellung bedeutet jedoch nicht zwangsläufig, daß der Apple-Benutzer mit seinen Problemen allein gelassen wird und deshalb zurückstecken muß. Er sollte die nachstehend aufgeführten Hinweise berücksichtigen. Diese Tips lassen sich wie folgt gruppieren:

1. Hardware-Verbesserung anschaffen (Saturn-128K-Karte und Accelerator-Karte);

2. Software-Unterstützung anschaffen (Cruncher, Compiler und EMBER);

Abb. 1

(1)	(2) Stückz.	(3) DM/St.	(3) Total	(4) A Min/St.	(4) A Min	(4) B Min/St.	(4) B Min	(4) C Min/St.	(4) C Min	(4) D Min/St.	(4) D Min
1	2500	165,43	413575	8,058	20145	0,108	270	0,121	303	0,412	1030
2	5000	67,88	339410	0,000	0	1,458	7290	1,180	5900	0,092	460
3	10000	31,62	316200	0,000	0	0,433	4330	0,374	3740	0,064	640
4	2000	142,76	285520	0,000	0	0,136	272	0,149	298	5,918	11836
(5):	19500		1354705		20145		12162		10241		13966
Kapazität:					22400		12500		11000		16000
Schlupf: (Leerzeiten in Minuten)					2255		338		759		2034
Auslastung in %:					89,93		97,30		93,10		87,29

(1) = Produktbezeichnung, (2) = Ist-Absatz, (3) = Deckungsbeitrag,
(4) = Fertigungsstelle, (5) = Summe

Abb. 2

22400	8,058	0	0	0	1	0	0	0	0	0	0	0
12500	0,108	1,458	0,433	0,136	0	1	0	0	0	0	0	0
11000	0,121	1,180	0,374	0,149	0	0	1	0	0	0	0	0
16000	0,412	0,092	0,064	5,918	0	0	0	1	0	0	0	0
3500	1	0	0	0	0	0	0	0	1	0	0	0
9000	0	1	0	0	0	0	0	0	0	1	0	0
11000	0	0	1	0	0	0	0	0	0	0	1	0
2600	0	0	0	1	0	0	0	0	0	0	0	1
0	165,43	67,82	31,62	142,76	0	0	0	0	0	0	0	0

Abb. 3

```
SCHMALHANS GMBH
ERMITTLUNG DER OPTIMALEN LOESUNG
NACH GERH. NIEMEYER

OPTIMUM NACH 4 DURCHLAEUFEN GEFUNDEN:

BASISVARIABLE            WERT:

     1        =        2779.85
     2        =        4884.71
     7        =         440.71
     4        =        2315.19
     9        =         720.15
    10        =        4115.29
     3        =       11000.00
    12        =         284.81

DER GESAMTDECKUNGSBEITRAG BETRAEGT 1469488.02 DM!
```

Abb. 4

Optimallösung-Plan 1 (gerundet)

(1)	(2) Stückz.	(3) DM/St.	(3) Total	(4) A Min/St.	(4) A Min	(4) B Min/St.	(4) B Min	(4) C Min/St.	(4) C Min	(4) D Min/St.	(4) D Min
1	2780	165,43	459895	8,058	22401	0,108	300	0,121	336	0,412	1145
2	4885	67,88	331604	0,000	0	1,458	7122	1,180	5764	0,092	449
3	11000	31,62	347820	0,000	0	0,433	4763	0,374	4114	0,064	704
4	2315	142,76	330489	0,000	0	0,136	315	0,149	345	5,918	13700
(5):	20980		1469808		22401		12500		10560		15999
Kapazität:					22400		12500		11000		16000
Schlupf: (Leerzeiten in Minuten)					-1		0		759		1
Auslastung in %:					100,01		100,00		96,00		99,99

(1) = Produktbezeichnung, (2) = Planabsatz, (3) = Deckungsbeitrag,
(4) = Fertigungsstelle, (5) = Summe

3. Programme schneller machen (optimieren).

3.1. Saturn- und Accelerator-Karte

Die Saturn-128K-Karte besteht aus 8 Bänken mit je 16K und kann sowohl als Pseudo-Disk als auch im gemischten Dienst eingesetzt werden. Das bedeutet, daß Programmteile und/oder Felder (Arrays) in das Extended RAM ausgelagert und bei Bedarf wieder zurückgeholt werden können, während die Pseudo-Disk-Funktion nicht beeinträchtigt wird. Leider ist das Handbuch (3) für „Applefreaks" geschrieben, so daß der Controller eine Weile benötigen wird, ehe er diese vielen Möglichkeiten richtig ausnutzen kann.
Am einfachsten ist die Anwendung der 128K-Karte in Verbindung mit Visicalc, weil hier nur eine spezielle Disk gebootet und einmalig angepaßt werden muß. Die Accelerator-Karte wurde bereits im Peeker (Heft 1/84, S. 19) beschrieben und ihre außerordentliche Wirkung wird im Verlaufe dieses Berichtes noch sichtbar werden. (Redaktioneller Hinweis: Der Autor benutzt einen Apple II+ mit einer Accelerator II+; der erwähnte Testbericht bezog sich auf die Accelerator IIe.)

3.2. EMBER-Software und Compiler

Fachleute schlagen vor, möglichst viele Befehle in einer Zeile unterzubringen. Diese Aufgabe erfüllt z.B. der **Dakin5-Cruncher** (4) sehr gut. Die Bedienung ist einfach: Man bootet die Dakin5-Diskette und wählt aus einem Menü die 12 aus. Nach ca. 10 Sekunden erscheint die in **Abb. 6** wiedergegebene Anleitung auf dem Bildschirm.
Ich besitze sowohl den **Hayden+-Compiler** (5) als auch den **Tasc-Compiler** (6). Normalerweise benutze ich für meine Arbeit den Hayden+, da er noch einfacher zu bedienen und vor allem sehr viel schneller ist, was die Vorbereitungszeit (Compiletime) und die Verarbeitungszeit (Runtime) angeht.
So benötigt der Tasc-Compiler z.B. für eine Optimierungsaufgabe des Umfangs 27 Zeilen und 47 Spalten mit einem BASIC-Programm des Autors Frazer (7) insgesamt 250 Sekunden Compiletime + 195 Sekunden Runtime = 445 Sekunden Gesamtzeit bei Einsatz der Accelerator-Karte. Der Hayden + -Compiler benötigt dagegen nur 31 Sekunden an Compiletime und 102 Sekunden an Runtime = 133 Sekunden Gesamtzeit. Allerdings ist das vom Hayden+ erzeugte Binärprogramm erheblich länger als das des Tasc-Compilers.

Abb. 5

Optimumlösung-Plan 2 (gerundet)

(1)	(2) Stückz.	(3) DM/St.	(3) Total	(4) B Min/St.	(4) B Min	(4) C Min/St.	(4) C Min	(4) D Min/St.	(4) D Min
2	5073	67,88	344365	1,458	7396	1,180	5986	0,092	467
3	11000	31,62	347820	0,433	4763	0,374	4114	0,064	704
4	2506	142,76	357757	0,136	341	0,149	373	5,918	14831
(5):	18579		1049942		12500		10474		16001
Kapazität:					12500		11000		16000
Schlupf: (Leerzeiten in Minuten)					0		526		-1
Auslastung in %:					100,00		95,21		100,01

(1) = Produktbezeichnung, (2) = Planabsatz, (3) = Deckungsbeitrag,
(4) = Fertigungsstelle, (5) = Summe

Abb. 6

```
            ** THE CRUNCHER **

(C) 1979 BY DAKIN5 CORPORATION

1. LOAD THE PROGRAM TO BE COMPRESSED

2. CALL 36361 TO RUN THE CRUNCHER

3. SAVE THE COMPRESSED PROGRAM

4. CALL 36402 TO RETURN TO PROGRAMMING
   AIDS 3.3 MENU
```

Abb. 7

Block	Autor	Acc.	Compile (sec)	Run (sec)	Total (sec)
1	Niemeyer optimiert	ja	25	22	47
	Niemeyer	ja	27	22	49
	Gillett	ja	27	44	71
	Bui	ja	48	73	121
	Frazer	ja	31	102	133
	Poole	ja	37	73	110
2	Niemeyer optimiert		25	50	75
	Niemeyer		27	51	78
	Gillett		27	120	147
	Bui		48	213	261
	Frazer		31	321	352
	Poole		37	206	243
3	Niemeyer optimiert	ja			61
	Niemeyer	ja			75
	Gillett	ja			206
	Bui	ja			385
	Frazer	ja			502
	Poole	ja			551
4	Niemeyer optimiert				204
	Niemeyer				261
	Gillett				684
	Bui				1298
	Frazer				1708
	Poole				1874

Abb. 8

Autor	Block Nr. 4 (ohne Hilfsm.) (sec)	Block Nr. 1 (Acc. + Comp.) (sec)	Faktor
Niemeyer optimiert	204	47	4,34
Niemeyer	261	49	5,33
Gillett	684	71	9,63
Bui	1298	121	10,73
Frazer	1708	133	12,84
Poole	1874	110	17,03

EMBER (8) steht für „Extended Memory Interpreter" und kann für den Apple II, II+ und IIe verwendet werden, und zwar mit oder ohne Language Card. EMBER kann insgesamt 4 Megabytes verwalten. Laut Handbuch (9) kann der Anwender Felder (Arrays) dimensionieren, die bis zu 64K groß sind, so daß
a) Integer-Arrays bis zu ca. 32.000 Elemente,
b) Real-Arrays bis zu ca. 12.000 Elemente,
c) String-Arrays bis zu ca. 16.000 Elemente umfassen dürfen.
Dabei bleibt es dem Anwender überlassen, ob er die ganze 128K-Karte für EMBER reservieren möchte (insgesamt 166.397 Bytes Speichergröße) oder nur bestimmte Bänke. Definiert man nur 3 Bänke und die Language Card für EMBER-Zwecke, so stehen insgesamt 84.477 Bytes Speicherplatz zur Verfügung. Laut Handbuch plaziert EMBER die Variablen in das Extended RAM, während der Programmtext in die unteren 48K geschoben wird. Die Speicherbelegung kann aus dem Handbuch (10) entnommen werden.
Wie simpel die Benutzung von EMBER ist, zeigt die nachstehende Beschreibung zur Vorgehensweise:

1. Phantom-Null-Diskette (für Accelerator) in Drive 1 einlegen und Apple einschalten.
2. Nachdem das rote Licht am Drive erlöscht ist, „S" tippen.
3. Nachdem das rote Licht am Drive erlöscht ist, Phantom-Null-Diskette herausnehmen und die EMBER-Kopie in Drive 1 einlegen. Dann die Leertaste drücken.
4. EMBER meldet sich mit einigen Erläuterungen und dem folgenden Menü:

(1) Ember set-up
(2) Install EMBER
(3) Make back-up copy

„1" <Return> eingeben. Es erscheint das nächste Menü:

(1) Look at Ember memory set-up
(2) Set-up EMBER memory usage
(3) Install Ember
(4) Exit

„2" <Return> eingeben. Menü Nummer 3 erscheint mit einer Inhaltsübersicht Slot/Bank. Darunter erfolgt eine Abfrage:

(1) Change slot contents
(2) Exit
Which option?

„1" <Return> eingeben. Nun muß die Frage nach dem Slot beantwortet werden. Hier wird immer ein „0" <Return> erforderlich sein, da entweder eine Language Card oder ein anderes Memory Board in Slot 0 stecken muß. Da mein Apple II+ sowohl mit einer Language Card als auch mit einem Saturn-128K-Board bestückt ist, tippe ich hier eine „0" und drücke die Return-Taste.
Wiederum erfolgt eine Abfrage:

(1) 128K RAM card (3) 32K RAM card
(2) 64K RAM card (4) 16K RAM card
What is now in the slot?

Dabei blinkt der angesprochene Slot 0. Mit „4" <Return> wird in Slot 0 der Besitz einer 16K-RAM-Card vermerkt. Die Frage: „Do you want to use the entire card?" wird mit „Y" <Return> beantwortet und das Programm zeigt wieder die Abfrage:

(1) Change slot contents
(2) Exit

„1" <Return> eingeben, wenn „Which option?" erscheint. Die Frage nach dem gewünschten Slot „Which slot (8=aux)?" wird mit „4" <Return> beantwortet, weil meine 128K-RAM-Karte in Slot 4 steckt.
Das bereits bekannte Menü

(1) 128K RAM card (3) 32K RAM card
(2) 64K RAM card (4) 16K RAM card
What is now in the slot?

wird mit „1" <Return> beantwortet. Die Frage „Do you want to use the entire card?" hat ihre Berechtigung. Denn wenn man jetzt „Y" <Return> tippt, dann werden in Slot 4 alle 8 Bänke der 128K-RAM-Card für EMBER reserviert, und es stehen insgesamt 166.397 Bytes zur Verfügung. Antwortet man aber mit „N" <Return>, wird jede einzelne Bank abgefragt und das Ergebnis in Slot 4 vermerkt.
Das folgende Menü

(1) Change slot contents
(2) Exit

wird mit „2" <Return> beantwortet, wodurch sich Drive 1 in Bewegung setzt. Nachdem das rote Licht am Drive erlöscht ist, steht das folgende Menü auf dem Bildschirm:

(1) Look at Ember memory set-up
(2) Set-up Ember memory usage
(3) Install Ember
(4) Exit

Mit „3" <Return> wird der Installationsprozeß abgeschlossen und die Anzeige „Ember installed" zeigt dies auch.

5. RUN Programmname <Return> startet dann das eigentliche Programm.

EMBER gestattet das Anlegen einer Turnkey-Version, so daß der eben beschriebene Vorgang sich auf das Booten der Phantom-Null- und der Turnkey-Diskette beschränkt. Dann kann sofort das gewünschte Programm gefahren werden.
EMBER zeichnet sich durch eine ausgesprochen einfache Bedienungsweise und eine enorme Leistungsfähigkeit aus. So braucht sich der Benutzer keinerlei Gedanken über das Bank-Switching zu machen, denn EMBER nimmt ihm derlei Arbeit ab.
Programme dieser Art sind Anwendern zu empfehlen, die nicht programmieren, sondern den Apple II+ als Arbeitswerkzeug benutzen wollen.
Bei den nachfolgenden Anwendungen hat sich herausgestellt, daß die Arbeitsgeschwindigkeit nicht von der Anzahl der von EMBER benutzten Bänke abhängt.

3.3. Applesoft-Verbesserung

Die unter (11) bis (16) aufgeführte Literatur empfiehlt für Applesoft-Programme u.a.:
1) Variablen anstatt Fließkomma-Konstanten benutzen.
2) Soweit als möglich sollte man Werte vor Beginn einer Schleife definieren.
3) FOR-NEXT-Schleifen sollten ohne Schleifen-Variablennamen enden, also statt NEXT I sollte man nur NEXT schreiben.
4) Unterprogramme und Zieladressen von GOTO-Befehlen sollte man an den Programmanfang verlegen.
5) Variablen sollten zu Programmanfang initialisiert werden.
6) Zuweisungen sollte man mit dem LET-Befehl versehen.
7) IF-THEN-Abfragen sollte man möglichst lange „falsch" sein lassen.
8) Möglichst kleine Zeilennummern verwenden.
9) REM-Zeilen weglassen.
10) Möglichst viele Befehle in einer Zeile unterbringen (Crunchen).

Ich habe einige dieser Vorschläge auf ihre Wirkung bei meinem Simplex-Optimierungsprogramm Niemeyer (17) überprüft, soweit sie nicht schon berücksichtigt waren. Das Programm Niemeyer benötigt in Verbindung mit der Accelerator-Karte 74 Sekunden für eine Matrix A (27,47). Die Rechenzeit kann um 11 Sekunden auf nunmehr 63 Sekunden gesenkt werden, wenn die Vorschläge Nr. 3 und Nr. 6 bei der Codierung berücksichtigt werden. Die Wirkung ist in etwa 50 zu 50. Wenn man das Weglassen des Schleifen-Variablennamens im optimierten Programm rückgängig macht, kann man genau ersehen,

daß das Programm nunmehr ca. 5,5 Sekunden (Stoppuhr) mehr benötigt.
Die vorgeschlagene Initialisierung der Variablen direkt zu Programmbeginn führte zu einer Verringerung des Ausführungsgeschwindigkeit; benötigte der Apple bisher nur 63 Sekunden, so waren es jetzt auf einmal 70 Sekunden. (Anm.d.Red.: Wenn einfache Variablen nach der Dimensionierung von Arrays angelegt werden, so müssen alle Felder im Speicher nach oben verschoben werden, da einfache Variablen stets unterhalb der Felder liegen.) Auch die Wirkung des Crunchens konnte ermittelt werden. Das Ausgangsprogramm brauchte nach dem Crunchen nur noch 65 Sekunden, also immerhin eine Verbesserung um 9 Sekunden. Wird das bereits erwähnte Programm nach der Optimierung (Vorschläge Nr. 3 und Nr. 6) gecruncht, so zieht dies eine weitere Abnahme der Rechenzeit auf 61 Sekunden nach sich.

4. Geschwindigkeitsvergleiche

Als ich die erste Optimierungsaufgabe in der Praxis lösen mußte, war kein BASIC-Programm verfügbar. Das von mir daraufhin erstellte Optimierungsprogramm benötigte für eine Matrix A (27,47) immerhin 2 volle Stunden. Das Warten auf die optimale Lösung bewog mich zu ausgedehnten Suchaktionen in Universitätsbibliotheken und Computershops.
Die nachfolgend unter (17) bis (21) näher bezeichneten Simplex-Programme haben eines gemeinsam, nämlich den Aspekt der Anschaulichkeit, während dabei die Verarbeitungsgeschwindigkeit in den Hintergrund tritt. Die Tabelle in **Abb. 7** beweist, daß erhebliche Zeitunterschiede in der Bearbeitung einer Matrix der Größe 27 Zeilen und 47 Spalten (Standardproblem Maximierung) zu verzeichnen sind, je nachdem welches Simplex-Optimierungsprogramm ausgewählt wird. Weiterhin wird die Wirkung von Compilern (hier Hayden+) und der Accelerator-Karte deutlich. Vergleicht man nun die Blöcke 4 und 1 miteinander, wie in **Abb. 8** geschehen, dann kann man am Faktor die Geschwindigkeitsverbesserung je Programm ablesen. Auch die Wirkung von Compiler bzw. Accelerator-Karte kann isoliert werden.
Beträgt die Matrixgröße etwa 70 Zeilen und 120 Spalten, dann genügt selbst ein DOS-Mover nicht mehr, um den benötigten Speicherplatz zur Verfügung zu stellen. In einem derartigen Fall wird man mit der 128K-Karte von Saturn gute Ergebnisse erzielen. Am einfachsten ist die Anwendung des EMBER-Programms, denn nach dem Starten der Phantom-Null-Diskette (wegen der Accelerator-Karte) und der bereits erwähnten EMBER-Turnkey-Diskette stehen immerhin max. 166.397 Bytes zur Verfügung, wobei allerdings bedacht werden muß, daß EMBER Felder (Arrays) mit max. 64K erlaubt.
Wählt man das Simplex-Programm von Poole aus, so liegt die Lösung erst nach ca. 17 Stunden und 57 Minuten vor. Dagegen benötigt der Apple unter den gleichen Voraussetzungen mit dem Programm von Gillett nur noch 8 Stunden und 7 Minuten. Das normale, also nicht optimierte Niemeyer-Programm wirft nach 1 Stunde und 59 Minuten die optimale Lösung aus, und das optimierte Niemeyer-Programm braucht nur 1 Stunde und 37 Minuten unter den gleichen Umständen. Würde man beim optimierten Niemeyer-Programm auf die Accelerator-Karte verzichten, so würde man ca. 5 Stunden und 22 Minuten warten müssen (Faktor 3,32).

5. Fazit

Dem Unternehmenscontroller steht heute eine Reihe von Möglichkeiten offen, um „als betriebswirtschaftlicher Begleiter für das Management" (22) noch effizienter tätig werden zu können als in der Vergangenheit. Der Apple II+ oder IIe bietet mit seinen Erweiterungen die Chance, wirkliches „Gewinnmanagement" (23) zu betreiben.

Literaturverzeichnis

(1) Deyhle, Dr. Albrecht, Controller Handbuch, Management Pockets 11, S. 123, 2. überarb. u. erw. Aufl. 1980, Management Service Verlag, Gauting/München.
(2) Schwarz, Prof. Dr. Horst, Kostenträgerrechnung und Unternehmensführung, Kapitel Bestimmung eines optimalen Produktionsprogramms, S. 88-97, 2. Aufl. 1973, Verlag Neue Wirtschafts-Briefe, Herne/Berlin.
(2a) Ich habe die Daten übernommen und die Tabellen der S. 88 entsprechend angepaßt.
(2b) Zitat von S. 96.
(3) Saturn System, Ann Arbor, USA.
(4) Dakin5 Corp., Denver, USA.
(5) Hayden's Basic Compiler + (hier Hayden+-Compiler genannt), Hayden Software, Lowell, USA.
(6) Microsoft Tasc The Applesoft Compiler, Microsoft, Bellevue, USA.
(7) Frazer, Ronald J., Angewandte lineare Programmierung, S. 69-70, Verlag Berlin Union, Stuttgart. (Ich habe das Simplex-Programm von Fortran in Applesoft umgeschrieben.)
(8) EMBER 1983, Micro Solutions; 1983, Titan Technologies.
(9) EMBER-Manual, S. 1, (von mir übersetzt).
(10) EMBER-Manual, S. 6.
(11) Feichtinger, Herwig, Effiziente BASIC-Programmierung, S. 23-24 aus Hobby-Computer 2, Funkschau Sonderheft Nr. 30, 3. überarb. Aufl., Franzis Verlag, München 1980.
(12) Applesoft Basic Programming Reference Manual, S. 120, Appendix E: Speeding up your program.
(13) Bartley, David / The Apple-Dillo, Program Optimization, A Call A.P.P.L.E. Tutorial, S. 9-12, Call A.P.P.L.E., Mai 1982.
(14) Thomas, Dieter, Schnellere Basic-Programme, MC, S. 32-33, Heft 7/1982, Franzis Verlag, München.
(15) NN, Optimieren von Programmen. Dem Rechner die Sporen geben, Chip, Heft 12, 1981, Vogel Verlag, Würzburg.
(16) Wagner, Roger, Speeding in Applesoft, S. 32-33, Call A.P.P.L.E. In Depth Number One, All About Applesoft.
(17) Niemeyer, Dr. Gerhard, Einführung in die lineare Planungsrechnung mit ALGOL- und FORTRAN-Programmen, Verlag Walter de Gruyter, Berlin 1968.
(18) Ich habe das „FORTRAN-Programm für Standard-Maximum-Probleme (reguläre Simplex-Methode)" auf S. 200-201 und 232-233 dargestellt, in Applesoft-BASIC übersetzt und eine optimierte BASIC-Fassung geschrieben.
(19) Gillett, Ph.D., Billy E.; Introduction to Operations Research, S. 100-105, McGraw-Hill, New York 1976.
(20) Bui, X.T., Kapitel 2 Lineare Programmierung: Die Simplex-Methode, S. 21-32 aus Planen und Entscheiden mit BASIC, 1. Aufl. 1983 Sybex-Verlag, Düsseldorf. (Eine sehr anschauliche und besonders empfehlenswerte Einführung in die Simplex-Methode und ihre BASIC-Programmierung!)
(21) Poole, Lon/Borchers, Mary, Some Common BASIC Programs, 3. Aufl., S. 103-107, Verlag Osborne/McGrawhill, Berkeley, USA.
(22) Deyhle, Dr. Albrecht/Bösch Martin, Arbeitshandbuch Gewinnmanagement, S. 12, Verlag Moderne Industrie 1979.
(23) ebenda, S. 13.
(24) Fa. Pandasoft, Berlin, bietet genauso wie (25) EMBER an, EMBER kostet lt. Preisliste vom 1.7.85 DM 198,-.
(25) Fa. Weiss, Wilhelmshaven.

Hinweis zum 2. Teil

Als zweiter und abschließender Teil ist ein Beitrag zum Thema „Matrizenrechnung in der betriebswirtschaftlichen Praxis" vorgesehen, der voraussichtlich im März-Heft erscheinen wird.

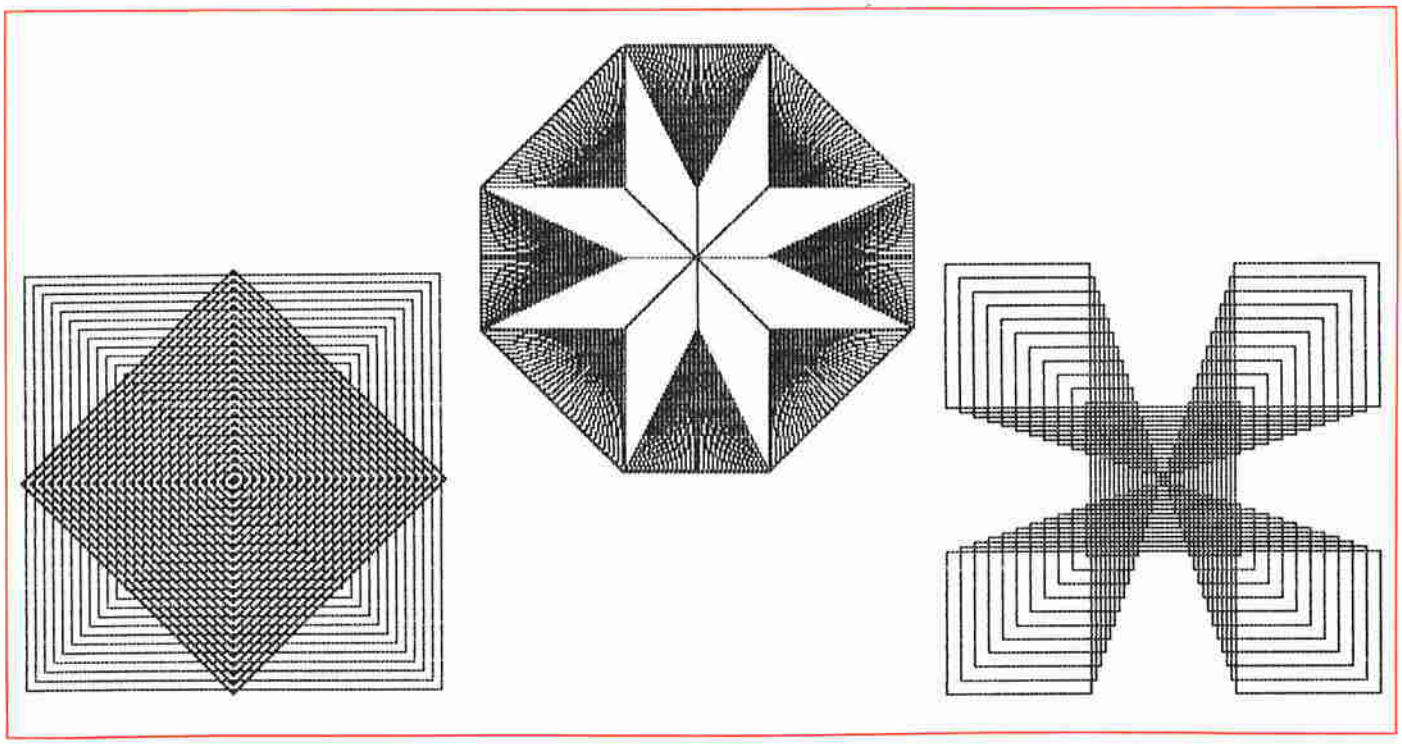
Grafik-Demonstrationen

Teil 2 von Ralf Knoke

Das nachfolgend abgedruckte Programm **GRAFIK.DEMOS.2** ist die Fortsetzung von „Grafik-Demonstrationen" aus Peeker, Heft 8/85, Seite 67. Der Zweck ist der gleiche geblieben. Es wird dargestellt, wie mit möglichst geringem Programmieraufwand möglichst effektvolle HGR-Grafiken entstehen. Wesentlichster Unterschied zu Teil 1 ist, daß die Pot-Punkte in den einzelnen Abschnitten nicht mehr direkt als feste Zahlenwerte, sondern alle als Variablen angesteuert werden. Alle Plot-Variablen stehen am Ende des Programms. Die Variablen erscheinen sinnvoll, da praktisch alle Werte mehrmals verwendet werden. Alles übrige ist gleich geblieben, ausgenommen natürlich die Grafiken selbst. Dadurch entfallen diesmal auch alle weiteren Erklärungen und es ist nur noch viel Spaß zu wünschen!

Kurzhinweise

1. Zweck:
Demonstration effektvoller Grafiken
2. Konfiguration:
II+, IIe oder IIc;
DOS 3.3 oder ProDOS
3. Test:
RUN GRAFIK.DEMOS.2
4. Sammeldisk:
GRAFIK.DEMOS.2
(Applesoft-Programm)



GRAFIK.DEMOS.2

```
1000 REM Ralf Knoke
1010 REM Juli 1985
1020 :
1030 REM ********* Teil 1a **********
1040 GOSUB 2650
1050 HGR2 : HCOLOR= 7:AZ = 3:CZ = 32
1060 IF BZ = 2 THEN AZ = 1: HCOLOR= 0:CZ = 96
1070 FOR X = 1 TO CZ
1080 HPLOT A,B TO A1,C: HPLOT D,B TO D1,C: HPLOT D,B1 TO
     D1,E: HPLOT A,B1 TO A1,E
1090 B = B - AZ::A1 = A1 + AZ:D1 = D1 - AZ:B1 = B1 + AZ
1100 NEXT X
1110 IF BZ = 2 THEN HPLOT G,E: GOTO 2600
1120 REM ********* Teil 1b **********
1130 GET A$
1140 GOSUB 2650
1150 FOR X = 1 TO 96:
1160 HPLOT A,B TO A1,C: HPLOT D,B TO D1,C: HPLOT D,B TO
     D1,E: HPLOT A,B TO A1,E
1170 A = A + 1:A1 = A1 + 2:D = D - 1:D1 = D1 - 2
1180 NEXT X
1190 REM ********* Teil 2 ***********
1200 GET A$
1210 GOSUB 2650
1220 HGR2 : HCOLOR= 7
1230 HPLOT G2,C TO G2,E: HPLOT C,B TO F,B
1240 FOR X = 1 TO 16
1250 HPLOT G2,C + 3 TO G + 8,B TO G2,E - 3 TO G1 - 8,B TO
     G2,C + 3
1260 C = C + 6:G = G + 8:E = E - 6:G1 = G1 - 8
1270 NEXT X
```

```
1280 REM ********* Teil 3 ***********
1290 GET A$
1300 BZ = 1
1310 HGR2 : HCOLOR= 7:AZ = 10
1320 GOSUB 2650
1330 FOR X = 1 TO 20
1340 IF BZ = 1 THEN HPLOT A,B TO G,AB TO AC,B
1350 IF BZ = 2 THEN HPLOT G,AB TO A,B TO G,H
1360 IF BZ = 1 THEN AB = AB + AZ
1370 IF BZ = 2 THEN A = A + AZ
1380 NEXT X
1390 BZ = BZ + 1: IF BZ = 2 GOTO 1320
1400 REM ********* Teil 4 ***********
1410 GET A$
1420 GOSUB 2650
1430 HGR2 : HCOLOR= 7
1440 HPLOT AE,C TO Z,C1 TO D1,Y1 TO D1,AH TO Z,E TO AE,E TO
     A1,AH TO A1,Y1 TO AE,C1 TO G,Y TO Z,C TO Z1,Y TO D1,Y1
     TO Z1,B TO D1,AH TO Z1,AI TO Z,E TO G,AI TO AE,E TO
     AG,AI TO A1,AH TO AG,B TO A1,Y1 TO AG,Y TO AE,C
1450 HPLOT G,Y TO G,AI: HPLOT Z1,Y TO AG,AI: HPLOT Z1,B TO
     AE,B: HPLOT Z1,AI TO AG,Y
1460 FOR X = 1 TO 32
1470 HPLOT G,Y TO AE + 2,C: HPLOT Z1,Y TO Z + 2,C1 + 2:
     HPLOT Z1,B TO D,Y1 + 2: HPLOT Z1,AI TO D1 - 2,AH + 2:
     HPLOT G,AI TO AE + 2,E: HPLOT AG,AI TO A1 + 2,AH + 2:
     HPLOT AG,B TO A,Y1 + 2: HPLOT AG,Y TO AF - 2,C1 + 2
1480 AE = AE + 2:Z = Z + 2:C1 = C1 + 2:Y1 = Y1 + 2:D1 = D1 -
     2:AH = AH + 2:A1 = A1 + 2:AF = AF - 2
1490 NEXT X
1500 REM ********* Teil 5 ***********
1510 GET A$
1520 GOSUB 2650
1530 HGR2 : HCOLOR= 7
1540 HPLOT A,C TO D,C TO D,E TO A,E TO A,C TO D,E: HPLOT D,C
     TO A,E
1550 FOR X = 1 TO 23:AZ = 4
1560 HPLOT A + AZ,C TO G - AZ,B1 - AZ TO A1,C1 + AZ: HPLOT D
     - AZ,C TO G1 + AZ,B1 - AZ TO D1,C1 + AZ: HPLOT D - AZ,E
     TO G1 + AZ,B + AZ TO D1,E1 - AZ: HPLOT A + AZ,E TO G -
     AZ,B + AZ TO A1,E1 - AZ
1570 A = A + AZ:G = G - AZ:B1 = B1 - AZ:C1 = C1 + AZ:D = D -
     AZ:G1 = G1 + AZ:B = B + AZ:E1 = E1 - AZ
1580 NEXT X
1590 REM ********* Teil 6 ***********
1600 GET A$
1610 GOSUB 2650
1620 HGR2 : HCOLOR= 7
1630 FOR X = 1 TO 12:AZ = 3
1640 HPLOT G,B TO R,S TO R,T TO D,B TO R,S TO R,T TO G,B TO
     U,V TO W,V TO G,C TO U,V TO W,V TO G,B TO Y,T TO Y,S TO
     A,B TO Y,T TO Y,S TO G,B TO W,Z TO U,Z TO G,E TO W,Z TO
     U,Z TO G,B
1650 S = S + AZ:T = T - AZ:U = U + AZ:W = W - AZ
1660 NEXT X
1670 REM ********* Teil 7 ***********
1680 GET A$
1690 GOSUB 2650
1700 HGR2 : HCOLOR= 7:AZ = 3
1710 HPLOT A,C TO A,B TO D,B TO D,E
1720 FOR X = 1 TO 28
1730 HPLOT D,B TO A1,B2 - AZ
1740 HPLOT A,B TO D1,B1 + AZ
1750 D = D - 7:B2 = B2 - 3:B1 = B1 + 3:A = A + 7
1760 NEXT X
1770 REM ********* Teil 8 ***********
1780 GET A$
1790 GOSUB 2650
1800 HGR2 : HCOLOR= 7:AZ = 4
1810 FOR X = 0 TO 69
1820 HPLOT C,C1 TO F,E TO F,C1
1830 IF X < 48 THEN HPLOT F1,C TO C1,E1 TO F1,E1
1840 C = C + AZ:F = F - AZ:E1 = E1 - AZ
1850 NEXT X
1860 REM ********* Teil 9 ***********
1870 GET A$
1880 HGR2 : HCOLOR= 7:AZ = 4:BZ = 1:CZ = 24
1890 GOSUB 2650
1900 FOR X = 1 TO CZ
1910 HPLOT G,B TO G1,B TO G1,B1 TO G,B1 TO G,B
1920 G = G - AZ:B = B - AZ:G1 = G1 + AZ:B1 = B1 + AZ
1930 NEXT X
1940 BZ = BZ + 1
1950 IF BZ = 2 THEN HCOLOR= 0: GOTO 1890
1960 IF BZ = 3 THEN HCOLOR= 7:CZ = CZ * 2:AZ = AZ - 2: GOTO
     1890
1970 REM ********* Teil 10 **********
1980 GET A$
1990 HGR2 : HCOLOR= 7:AZ = 4:BZ = 1:CZ = 1
2000 GOSUB 2640
2010 FOR X = 1 TO 1
2020 HPLOT G,B TO G1,B TO G1,B1 TO G,B1 TO G,B
2030 HPLOT G2,B - 2 TO G1 + 2,B2 TO G2,B1 + 2 TO G - 2,B2 TO
     G2,B - 2
2040 NEXT X
2050 BZ = BZ + 1
2060 IF CZ > 23 GOTO 2110
2070 IF BZ = 2 THEN HCOLOR= 0: GOTO 2010
2080 IF BZ = 3 THEN HCOLOR= 7: GOTO 2010
2090 IF BZ = 4 THEN BZ = 1: HCOLOR= 7:G = G - AZ:B = B - AZ:
     G1 = G1 + AZ:B1 = B1 + AZ:CZ = CZ + 1: GOTO 2010
2100 REM ********* Teil 11 **********
2110 GET A$
2120 HCOLOR= 7
2130 GOSUB 2650
2140 FOR X = 0 TO 191
2150 HPLOT C,C1 TO F,C1
2160 HCOLOR= 0: HPLOT C,C1 TO F,C1
2170 C1 = C1 + 1
2180 HCOLOR= 7: NEXT
2190 REM ********* Teil 12 **********
2200 GET A$
2210 GOSUB 2650
2220 HGR2 : HCOLOR= 7:AZ = 2
2230 FOR X = 1 TO 96
2240 HPLOT A,C TO D,E: HPLOT D1,C1 TO A1,E1
2250 A = A + AZ:D = D - AZ:C1 = C1 + AZ:E1 = E1 - AZ
2260 NEXT
2270 REM ********* Teil 13 **********
2280 GET A$
2290 GOSUB 2650
2300 HGR2 : HCOLOR= 7
2310 FOR X = 1 TO 32
2320 HPLOT C,C TO C1,B TO G,C: HPLOT C,E TO C1,B TO G,E:
     HPLOT F,C TO C1,B TO F,E
2330 C1 = C1 + 9: NEXT
2340 REM ********* Teil 14 **********
2350 GET A$
2360 GOSUB 2660: HGR2 : HCOLOR= 7
2370 FOR X = 1 TO 16
2380 BZ = AL - AJ
2390 HPLOT AJ,AK TO AJ - BZ,AK TO AJ - BZ,AK - BZ TO AJ,AK -
     BZ TO AJ,AK TO AL,AK TO AL + BZ,AK TO AL + BZ,AK - BZ
     TO AL,AK - BZ TO AL,AK TO AL,AM TO AL + BZ,AM TO AL +
     BZ,AM + BZ TO AL,AM + BZ TO AL,AM TO AJ,AM TO AJ -
     BZ,AM TO AJ - BZ,AM + BZ TO AJ,AM + BZ TO AJ,AK
2400 AJ = AJ - 2:AK = AK - 2:AL = AL + 2:AM = AM + 2
2410 NEXT
2420 REM ********* Teil 15 **********
2430 GET A$
2440 GOSUB 2650: HGR2 : HCOLOR= 7
2450 HPLOT C,AN TO G,AO TO F,AN
2460 FOR X = 1 TO 34
2470 HPLOT C + 4,AN - 4 TO G + 4,AO + 4
2480 C = C + 4:AN = AN - 4:G = G + 4:AO = AO + 4
2490 NEXT
2500 REM ********* Teil 16 **********
2510 GET A$
2520 GOSUB 2650: HGR2 : HCOLOR= 7
2530 HPLOT C,B TO AP,AR TO AQ,AS TO F,B TO AQ,AR TO AP,AS TO
     C,B
2540 FOR X = 1 TO 17
2550 HPLOT C,B TO AP,AR TO G,B TO AQ,AR: HPLOT C,B1 TO AP,AS
     TO G,B1 TO AQ,AS
2560 C = C + 4:AP = AP + 4:B = B - 4:AR = AR + 4:G = G + 4:
     B1 = B1 + 4:AS = AS - 4:AQ = AQ + 4
2570 NEXT
2580 REM ********* Teil 17 **********
2590 GET A$
2600 HOME : TEXT : VTAB (10): HTAB (5): PRINT "BITTE
     WAEHLEN:": PRINT : PRINT "(1) NOCH EINMAL?": PRINT :
     PRINT "(2) ENDE"
2610 PRINT : GET A$
2620 IF A$ = "1" OR A$ = "J" GOTO 1040
2630 HOME : END
2640 REM ********* Initialisierung ***
2650 A = 43:A1 = 43:B = 95:B1 = 95:B2 = 95:B3 = 95:C = 0:C1
     = 0:D = 235:D1 = 235:E = 191:E1 = 191:F = 279:F1 = 279:
     G = 139:G1 = 139:G2 = 139:H = 190
2660 R = 215:S = 60:T = 130:U = 104:V = 19:W = 174:Y = 63:Y1
     = 63:Z = 171:Z1 = 171:AB = 1:AC = 234:AE = 107:AF =
     107:AG = 107:AH = 127:AI = 127:AJ = 138:AK = 94:AL =
     140:AM = 96:AN = 165:AO = 26:AP = 69:AQ = 208:AR = 25:
     AS = 165
2670 RETURN
```

Zeile 2390 muß ohne Leertasten eingegeben werden, da sie ansonsten nicht in den Eingabepuffer paßt.

Video-Clips für jedermann

Die Video-1000-Karte

Erfahrungsbericht von Norbert Man Brandl

Ein neues Bildgefühl

Wenn bis zu 512 digitalisierte Fernseh- oder Videobilder auf dem Monitor eines Apple II das Laufen lernen und dabei ein Tempo von bis zu 13 Bildern pro Sekunde erreichen, dann steckt des Rätsels Lösung nicht in der „neuen" Firmenpolitik des Konzerns mit dem gestreiften Apfel auf dem Surfsegel, sondern im Slot 7 der immer noch aktuellen Wozniak-Maschine. Für diesen Slot wurde im Berliner Ingenieur-Büro M. Fricke eine neue Digitalisierungskarte entwickelt, die sich durch zwei besondere Qualitäten von allen bisherigen Karten dieser Art unterscheidet:

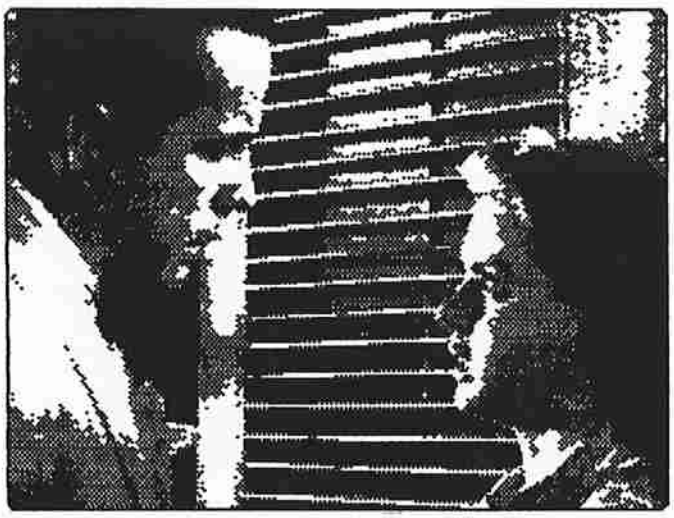
1. Das von der Karte empfangene Videosignal wird in 384 x 320 Bildpunkten digitalisiert, also 16K Bildinformation, was erheblich mehr ist, als sonst in der Apple-Klasse geboten wird und jeglicher weiteren Bearbeitung eine solide Ausgangsbasis verschafft.
2. Das Video-1000-Interface unterstützt von sich aus RAM-Karten bis 1 MByte; eine Fähigkeit, die erstaunliche Möglichkeiten eröffnet, wie z.B. serielle Digitalisierungsvorgänge und Ganzbild-Animation.

Kabelsalat als Vorspeise

Bis die Karte allerdings ihre Vorzüge dem Benutzer unter Beweis stellen kann, lernt dieser erst einmal ihre Unzulänglichkeiten kennen. Auf der kleinen, sehr sauber verarbeiteten Interfaceplatine sind 3 Cinch-Eingänge so dicht nebeneinander montiert, daß die im Videobereich üblichen Cinch-Stecker mit Metallummantelung nicht genügend Platz finden. In eine derart engstirnige Raumaufteilung lassen sich lediglich plastikumfaßte Billigstecker in Reih und Glied bringen. Doch welche Kamera, welcher Videorecorder verfügt zur Zeit noch über solche zierlichen Zuführungen? Mir ist klar, daß dem Normen-Dschungel der Videohersteller nicht mit einfachen Mitteln Paroli zu bieten ist, doch sollte man dessen Existenz nicht völlig außer acht lassen. Deswegen lautet mein Vorschlag an die Video-1000-Entwickler: Cinch-Eingänge nicht direkt auf die Platine löten, sondern durch kurze Zuleitungen bis an die Rückwand des Computers herausführbar gestalten. Außerhalb des Computers ist jedenfalls mehr Platz für dicke Videokupplungen und Kabel.

Obwohl ich von Berufs wegen seit Jahren mit Video arbeite und über eine beträchtliche Anzahl verschiedener Kabel und Stecker verfüge, eilte ich also, mit dem Ladenschlußgesetz als unsichtbarem Damoklesschwert über dem Haupt, durch die Videoabteilungen des Fachhandels, um mir die zierlichsten Cinch-Stecker, die auffindbar waren, zu besorgen.

Zwei Stunden später und fünfzig Mark ärmer war es mir endlich gelungen, die Video- und die Computerwelt zufriedenstellend zu verbinden. Drei Kabel fanden sich nach einigem Widerstand mit der Enge im Inneren des Computers ab. Das eine war über Adapter und Verlängerung mit einem Videorecorder verbunden. Das zweite war bereit, das Videosignal aus dem Apple aufzunehmen, es durch die Karte zu schleifen und über das dritte Kabel an den Monitor abzugeben. Dieses auf den ersten Blick etwas umständlich wirkende Verbindungssystem befreit von dem Zwang eines



Now it's looking at you, kid.

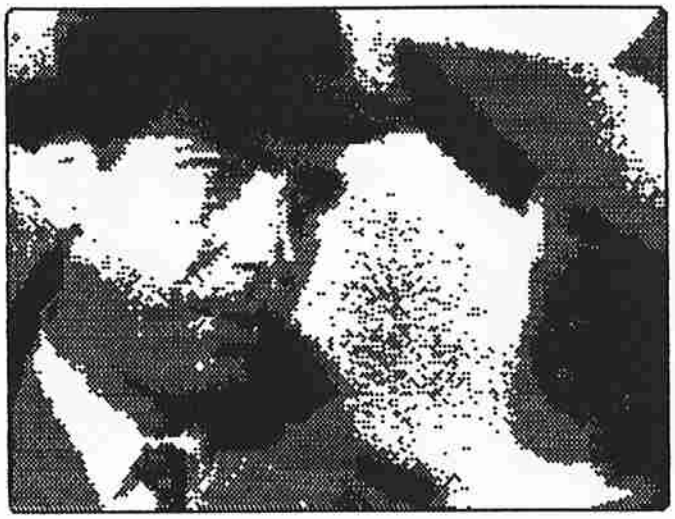
Remember Paris?

zweiten Kontrollmonitors und läßt den Computer ohne zusätzliche Schalt- oder Umstöpselmanöver seine sonstigen Anwendungen wie gewohnt abarbeiten.

Anm.d.Red.: Die Firma Hamaphot KG, Monheim (Bayern), hat uns ihren neuen 640seitigen Farbkatalog „hama photo + video. Das große Neuheitenprogramm im Zubehör-Ratgeber 1985/86" zugesandt, der für eine Schutzgebühr von DM 10,- erhältlich ist und u.a. seitenweise Cinch-Stecker aller Art und Größe beschreibt.

Euphorie und Bilderfieber

Nach einem letzten, flüchtigen Blick auf die zwei beidseitig mit Matrixdruck gestalteten DIN-A4-Blätter, die, nicht ohne den Leser mit einigen köstlichen Tippfehlern zu unterhalten, in knapper und sachlicher Form über Bedienung und Möglichkeiten der Karte Auskunft geben, bootete ich die mitgelieferte Diskette und erlebte, wie schnell sich anfänglicher Ärger in nachhaltige Zufriedenheit verwandeln kann.
Software und Karte bilden zusammen ein verblüffend schnelles Werkzeug. Die Bedienungsschnittstelle ist kinderleicht zu handhaben. Sobald die Frage nach einer etwaigen RAM-Karte negativ beantwortet ist, liegt das Videosignal ohne Grauwerte sofort auf dem Monitor vor. Andernfalls müssen Slot und Größe der RAM-Erweiterung dem Programm mitgeteilt werden. Mit den Pfeiltasten ist die Helligkeit des Videobildes einstellbar, wodurch selbst kontrastarme oder schlecht ausgeleuchtete Videovorlagen in den Griff zu bekommen sind. Eine fest implementierte Alternative zu dem normalen Schwarz-Weiß-Strichbild stellt das Punktrasterbild dar, das über die Taste „P" anwählbar ist. Die Rasterung bietet eine einfache Art, Grauwerte vorzutäuschen. Ob nun im Raster- oder Strichmodus, ein Druck auf die Taste „R" digitalisiert das Bild und schreibt es sowohl in das Video-1000-RAM als auch in den Hauptspeicher des Computers, von $4000-$7BFF. Weil demzufolge der Bereich der zweiten hochauflösenden Seite belegt ist, wird ein Ausschnitt des digitalisierten Bildes über die erste HGR-Seite angezeigt. Dieser Ausschnitt ist mit Hilfe der vier Pfeiltasten (Apple II+: A = aufwärts, Z = abwärts) über das gesamte 16K-Bild „verschiebbar", indem die entsprechenden Bildteile in die HGR-Seite übertragen werden. Dieser Vorgang ist sehr schnell und macht die Wanderung über das digitalisierte Großbild zu einer vergnüglichen Entdeckungsreise. Besteht der Wunsch, das ganze Bild auf einen Blick zu sehen, kann ein Rasterbild mit „Ctrl-P" und ein normales Strichbild mit „Ctrl-N" auf die Dimensionen der HGR-Seite verkleinert werden. Sowohl die Ausschnitte als auch die Verkleinerungen sind in jeweils 34-Sektoren-Binärdateien auf Diskette speicherbar und für übliche Grafik- und Drucksoftware zugänglich. Die 16K-Bilder müssen hingegen mit „Ctrl-S" und einer Nummer von 1 bis 8 auf gesonderte Disketten geschrieben werden, die keinen „CATALOG" besitzen und maximal 8 ausschließlich über Nummern identifizierbare Bild-Dateien beinhalten. Die erste dieser großen Binärdateien belegt Track 3-6, die zweite Track 7-11 usw. bis Datei Nr. 8, die sich von Track 31 bis Track 34 erstreckt. Die Geschwindigkeit, mit der diese Jumbo-Files geschrieben und mit „Ctrl-L + Nummer" gelesen werden, hat mich beeindruckt.
Da weder Inhaltsverzeichnisse noch Bildernamen den Benutzer durch eine schnell wachsende Sammlung abgespeicherter Video-1000-Bilder lotsen, sollte der Griff zur klassischen Kombination aus Bleistift und Papier als Gedächtnisstütze nicht allzuspät erfolgen. Es ist übrigens die einzige Zettelwirtschaft, die tastaturverwöhnten Fingern abverlangt wird, denn sämtliche Befehlsfolgen und Informationen über den Status praesens des Systems sind über eine Hilfsseite einblendbar.
Wie nützlich die stets aktuellen Nachrichten der elektronischen Help-Page sein können, wurde mir erst bewußt, als ich zum ersten Mal eine 512K-RAM-Karte mit ins Video-1000-Spiel einbrachte. Der zusätzliche Speicherbereich, den man der Digitalisierungsarbeit zur Verfügung stellen kann, wird nämlich vom System aus genauso gehandhabt wie physische Disketten. Daraus ergibt sich einerseits die Schwelle von mindesten 128K auf einer RAM-Karte, um überhaupt für das Video-1000-System von Interesse sein zu können, und andererseits die Unterteilung der größeren RAM-Karten, die ja – wie eingangs erwähnt – bis in den 1MByte-Bereich hinein akzeptiert werden, in jeweils 128K-Blocks. Schon in der von mir getesteten Konfiguration standen also 4 Blocks für insgesamt 32 16K-Bilder zur Verfügung. Bewegen sich hierbei die Wünsche und Bedürfnisse des Anwenders auf der voll automatisierten Arbeitsebene, die das Video-1000-Interface bietet, braucht kein Gedanke über Blockzuweisung und Bildnummer verschwendet zu werden. Mit dem Befehl „Ctrl-R" digitalisiert die Karte selbständig Bilderfolgen in einem Zeittakt von 4 Bildern pro Sekunde, solange die



I stick my neck out for nobody.

The beginning of a beautiful friendship . . .

zugewiesene Speicherkapazität ausreicht. Das Abspielen der Bilderfolgen wird durch den Befehl „Ctrl-F" erreicht. Beides verwandelt den Apple in eine ungeahnt flinke Bildermaschine. Sobald aber Anwendungen zu bewältigen sind, die das Transferieren, Abrufen oder Einsortieren einzelner Bilder innerhalb der erweiterten Speicherstruktur unumgänglich machen, ist es notwendig, eine genaue Information über den gerade aktiven RAM-Block zu besitzen. Ob nun das Neuausrichten der Videokamera, das Suchen bestimmter Szenen auf dem Videoband oder andere notwendige Tätigkeiten den schnellen Arbeitsfluß, den die Karte ermöglicht, unterbrechen und die Konzentration ablenken – die Taste mit dem „?" hilft stets weiter. Ein Blick auf die Hilfsseite bringt den aktiven Speicherblock wieder in Erinnerung und unterstreicht die freundliche Qualität einer Help-Page, die ihre Aufgabe diskret und bedarfsgerecht erledigt, ohne die Arbeitsfläche eines kleinen 8-Bit-Systems mit Symbol-Girlanden und Rollmenüs zu belasten.

Und wie geht's weiter?

Bereits die Grundkonfiguration aus Interface und Software liefert genügend Argumente, die das Video-1000-System sowohl im Hobby- als auch im Profibereich hohem Anspruch gerecht werden läßt. Vollends aus dem Kreis der Mitbewerber hebt sich das System durch seine außerordentlich flexiblen Ausbaumöglichkeiten hervor. Das Ingenieur-Büro M. Fricke liefert ein breites Spektrum von RAM-Karten, die das beste mir bekannte Preis/Leistungsverhältnis auf dem deutschen Markt vorweisen können. Darüber hinaus sorgen die Berliner Hersteller auch für die softwaremäßige Zukunft ihres Systems. Drei weitere Software-Pakete werden ab November 1985 die Video-1000-Möglichkeiten potenzieren. Es handelt sich hierbei im einzelnen um

– ein Programm zur Digitalisierung von Video-Standbildern in bis zu 16 Graustufen und deren Ausgabe in hoher Matrixdruckqualität,
– eine Sammlung von Hires-Utilities zur komplexen Nachbearbeitung der digitalisierten Vorlagen und schließlich um
– RAM-Karten unterstützende Software, die aus laufenden Fernseh-, Videoband- oder Kamera-Aktionen bis zu 512 Bilder digital umwandelt und als Computer-Clips vorführt.

Das sensationellste an der Video-1000-Geschichte ist jedoch der Preis. Für nur ca. DM 300,- läßt das Video-1000-System die hellsten Weihnachtsglöckchen erklingen, und das völlig unabhängig von Kalendervorschriften.

Apple DOS 3.3 – Tips und Tricks

Die völlig überarbeitete dritte Auflage (mit neuer Begleitdiskette) erscheint Anfang 1986.

Dr. Alfred Hüthig Verlag · Postfach 10 28 69 · 6900 Heidelberg

The Write Choice

Ein preiswertes Textverarbeitungsprogramm

Erfahrungsbericht von Franz-Josef Hüskens

Für den Apple II gibt es inzwischen zahlreiche Textverarbeitungsprogramme. Das bekannteste dürfte wohl „Wordstar" sein. Mit einem Preis von über 1000,- DM schreckt es jedoch sicherlich viele Privatleute und Kleingewerbetreibende ab.
Von der Firma Apple selbst wird der „Applewriter" sowie die in „Appleworks" integrierte Textverarbeitung angeboten. Die hierfür verlangten Preise von ca. 500,- bzw. über 800,- DM sind auch nicht von Pappe. Voraussetzung für die Anschaffung von derart teuren Textverarbeitungs-Programmen ist eine kommerzielle und/oder häufige Nutzung. Für die gelegentliche Anwendung im Heimbereich oder in kleineren Betrieben muß jeder Anwender für sich selbst entscheiden, ob sich eine derart kostspielige Anschaffung lohnt.
Es gibt jedoch auch sehr preiswerte Textverarbeitungsprogramme, die fast das gleiche leisten. Eines dieser „billigen" Systeme soll hier vorgestellt werden.
Es handelt sich um **The Write Choice**, vertrieben von der Firma Roger Wagner, deren Inhaber langjährigen Apple-Besitzern kein Unbekannter sein dürfte: Von ihm stammen die Programmsammlung Apple-Doc und verschiedene andere Softwarepakete. Außerdem ist er als Buchautor hervorgetreten.

„Write Choice" besteht aus vier Teilen:
– Word Processing Style Manual (im 152seitigen Handbuch enthalten)
– The Analyst
– Tut's Typer
– The Correspondent (Hauptprogramm)

1. Word Processing Style Manual

Das „Word Processing Style Manual" ist Teil des Handbuchs und erklärt auf ca. 50 Seiten, wie man einen guten Stil schreibt, und gibt Tips für Geschäftsbriefe, Manuskripte usw. Da das Programm aus Amerika kommt, ist es auf die englische Sprache ausgerichtet. Man kann damit also sein eigenes Englisch auffrischen oder als Schüler seine Sprachkenntnisse vertiefen.

2. The Analyst

Der „Analyst" ist ein Programm, das die Lesbarkeit eines Textes untersucht. Dazu werden im Text verschiedene Werte, die für die Lesbarkeit relevant sind, festgestellt. Das Programm scheint jedoch nicht richtig zu arbeiten, da es manchmal sehr seltsame Resultate anzeigt. Dieser Text sollte einmal aus über 20.000 Wörtern bestehen, dann aus etwa 19 000. Die vermutlich (und hoffentlich) richtigen Werte wurden nach mehrmaligem Probieren angezeigt. Das Programm ist für meine Begriffe eine Spielerei mit wenig Nutzen.

3. Tut's Typer

„Tut's Typer" ist ein Hires-Grafik-Programm, das auf unterhaltsame Weise einen „Schreibmaschinenkurs" bietet und zusätzlich durch ein Spiel die Schreibfähigkeit trainiert:

a) Der erste Teil des Programms benutzt die Hires-Grafik-Fähigkeiten des Apple, um die Tastatur auf dem Bildschirm abzubilden. Anhand dieser Grafik-Tastatur wird die richtige Plazierung der Finger auf den Tasten geübt und mit verschiedenen Lektionen die richtige Bewegung der Finger auf der Tastatur gezeigt. Nach Beendigung jeder Lektion erhält man einen Ausdruck über die Anzahl der fehlerhaften Buchstaben bzw. der falschen Wörter. Gehört man bereits zu den fortgeschrittenen Maschinenschreibern, so wird dieses Programm schnell langweilig, weil es zu langsam ist. Schnelleres Schreiben ist wegen der zeitraubenden Darstellung der Tastatur auf dem Bildschirm unmöglich.

b) Der zweite Programmteil ist für fortgeschrittene Schüler gedacht. Hier kommt es auf Geschwindigkeit an. Man findet sich als „Spieler" in den Labyrinthen der Pyramide des Pharao Tut wieder. Im ersten Korridor des Labyrinths tauchen Wörter auf, die eingetippt werden müssen. Gelingt dies nicht, bevor ein Wort das Ende des Korridors erreicht hat, so verschwindet es wieder. Andernfalls taucht es in einem zweiten, später in einem dritten Korridor auf usw. Jedesmal muß das Wort vor seinem Verschwinden eingegeben werden. Auf den ersten Blick scheint dies kein Problem zu sein. Bei einem Wort! Es bleibt jedoch nicht dabei, denn in regelmäßigen Abständen tauchen immer mehr Wörter auf. Nach einer gewissen Zeit wird das „Spiel" beendet, das Ergebnis angezeigt und auf Diskette festgehalten. Mit diesem Programmteil können Wettkämpfe zwischen bis zu 25 Teilnehmern ausgetragen werden. Damit keine Langeweile aufkommt, können auch neue Lehrstücke aufgestellt werden. In dem 12seitigen Anhang des Handbuches wird erklärt, wie man neue Lektionen mit dem „Correspondent" entwickelt.

4. The Correspondent

Der „Correspondent" ist das „Herz" des Programmpakets. Es ist das eigentliche Textverarbeitungsprogramm mit folgenden Merkmalen:

Horizontales Scrollen: Der eingegebene Text wird auf dem Bildschirm so angezeigt, wie er auch später auf dem Papier zu sehen ist. Das Besondere dabei ist, daß die Anzeige sowohl mit der 80-Zeichenkarte als auch im 40-Z/Z-Modus möglich ist. Die maximale Zeilenlänge ist frei wählbar. Ohne 80-Zeichenkarte beträgt sie 80 Zeichen; mit entsprechender Hardware (9 80-Z/Z-Karten werden unterstützt) können maximal 160 Zeichen pro Zeile bearbeitet werden. Mit der Videx-Ultraterm-Karte kann eine Zeile angeblich bis zu 255 Zeichen lang sein! Ist die Zeile länger, als es die Anzeigekapazität zuläßt, wird trotzdem alles zeilenrichtig wiedergegeben.
Der auf dem Bildschirm angezeigte Ausschnitt bewegt sich entsprechend der Cursorposition in die linke bzw. rechte Hälfte des Textes. Mit Ctrl-O kann man sich – unabhängig von der momentanen Cursorposition – den nicht sichtbaren Textteil ansehen. Durch eine ständig eingeblendete **Statuszeile** wird man über die aktuelle Position des Cursors im Text informiert. Angezeigt wird die Seite, die Zeile und die Spalte, in der sich der Cursor befindet. Die Statuszeile gibt auch Auskunft über den Modus, in dem gerade gearbeitet wird (Edit bzw. File-Modus). Eine **Tabulatorzeile** wird am Anfang einer jeden Seite angezeigt. In dieser Zeile können mit Ctrl-S Tabulator-Stopp-Punkte beliebig gesetzt oder gelöscht werden. Ein Stern in der Mitte der Tabulatorzeile zeigt die tatsächliche Mitte der Textseite an.

Eingabe-Modus: Die Texteingabe erfolgt so, wie man es von anderen Textverarbeitungsprogrammen her gewohnt ist: Ein Drücken der Return-Taste ist nur am Ende eines Absatzes nötig. Falsche Buchstaben werden einfach durch Überschreibung berichtigt. Die Formatierung des Textes erfolgt „wort-richtig" direkt auf dem Bildschirm. Das Programm erkennt automatisch, wenn ein Wort nicht mehr in die laufende Zeile paßt, und springt dann selbsttätig zum Anfang der folgenden Zeile. Wird die Trennung von Worten gewünscht, so muß der Trennstrich von Hand eingegeben werden. Der „Correspondent" erkennt ihn am Zeilenende und formatiert entsprechend.

Im **Editier-Modus** kann der Cursor beliebig im Text bewegt werden. Die Tasten Ctrl-U, Ctrl-H, Ctrl-J und Ctrl-K erlauben neben den Pfeiltasten die Bewegung des Cursors nach rechts, links, unten und oben. Für die schnelle Überbrükkung größerer Entfernungen im Text kann man mit Ctrl-T zum nächsten Tabulator-Stopp springen. Ctrl-Y bewegt den Cursor zum nächsten Wort. Abhängig von der zuletzt gedrückten Pfeiltaste (rechts, links) sind diese „weiten Sprünge" in beide Richtungen möglich. Mit Ctrl-B springt man zum Textanfang, mit Ctrl-E zum Textende. Mit Ctrl-V kann man sich die letzte durchgeführte Änderung im Text anzeigen lassen.
Der „Correspondent" hat folgende Editier-Befehle: Fehlende Wörter werden mit Ctrl-I in den bestehenden Text eingefügt, falsche Wörter mit Ctrl-G gelöscht, falsche Zeichen mit Ctrl-R entfernt. Der letzte Befehl dient auch zur Formatierung des Textes. Ctrl-Z löscht den Text einer Zeile ab der Cursorposition.
Weitere Kommandos sind: Ctrl-C zur Zentrierung einer Zeile, Ctrl-D zur Duplizierung des zuletzt eingegebenen Zeichens. Mit Ctrl-F kann man Zeichenfolgen suchen und auch austauschen lassen. Die ESC-Taste wird bei älteren Apple-Modellen zur Eingabe von Großbuchstaben benutzt. Die Eingabe der deutschen Sonderzeichen (äöüß) in Groß- und Kleinschreibung ist bei den älteren Apple-Modellen normalerweise nicht möglich. Durch ein vorangestelltes Ctrl-A können auch diese Sonderzeichen eingegeben werden. Ctrl-A ermöglicht ferner die Eingabe von inversen (Ctrl-A I), blinkenden (Ctrl-A F) und normalen (Ctrl-A N) Zeichen. Damit lassen sich also auch Ctrl-Zeichen (blinkend) ein-

geben. Verschiedene Drucker werden damit steuerbar.

Die gleichen und noch weitere Befehle stehen auch im **File-Modus** zur Verfügung. Den File-Modus schaltet man mit Ctrl-X (aus dem Editier-Modus heraus) an. Die zwei wichtigsten Befehle sind G (Get a file = Text laden) und S (Save a file = Text speichern). Des weiteren ist es möglich, den Anfang und das Ende des Textes zu setzen (Ctrl-B, Ctrl-E). Mit Ctrl-P kann der Text ausgedruckt werden. Steht eine 80-Zeichenkarte zur Verfügung, so kann der Text mit der Tastenfolge ESC Ctrl-P auf dem Bildschirm „probegedruckt" werden. Vor dem Ausdruck kann man die Papierlänge, den linken Rand, die unterste Zeile usw. festlegen. Das Verschieben und Kopieren von Textblöcken ist ebenfalls möglich, ferner die Eingabe normaler DOS-Kommandos.

Weitere Merkmale des „Correspondent" sind: Eingegebener Text kann als Binär- oder als Textfile auf Diskette gespeichert werden. Da nur fünf bzw. – bei Verwendung der 64K-Karte – sieben Seiten Text im Speicher abgelegt werden können, gibt es die Möglichkeit, verschiedene Files untereinander zu einem großen Text zu verbinden. Diese Einzeltexte werden nicht nur hintereinander gedruckt, sondern können auch mit der Such-Routine kontinuierlich durchsucht werden. Sogar die Such- und Tausch-Routine arbeitet dann richtig, so daß nach einem Tausch erst der neue Text auf Diskette gespeichert und dann der neue File geladen wird. Das Zusammenfügen verschiedener Texte ist ebenso möglich wie das Speichern von Teilen des Gesamttextes.
Mit dem gesondert erhältlichen Software-Paket „The Printographer" kann man sogar Grafik in den Text einbinden.

Hilfe-Seiten dienen als Gedächtnisstütze, da man die vielen Möglichkeiten, die der Correspondent bietet, nicht alle auswendig wissen kann. Von Hause aus wird das Programm mit sieben Hilfe-Seiten geliefert. Zusätzlich wird jedoch im Handbuch beschrieben, wie man selbst derartige Seiten aufstellt und speichert.
Das mitgelieferte **Handbuch** erklärt die Benutzung anhand von Beispielen auf 60 Seiten sehr ausführlich. In den verschiedenen Anhängen wird zudem beschrieben, wie man den „Correspondent" modifizieren kann.

The Write Choice ist auf allen Apple-II-Typen (II+/e/c) lauffähig. Da der Ausbau des Computers unterschiedlich sein kann, wird auf der Diskette ein Programm mitgeliefert, mit dem man The Write Choice „personalisieren" kann.
Auf der Diskette befindet sich eine modifizierte Version des „Diversi-DOS". Dies macht sich besonders durch das erfreulich schnelle Laden und Speichern der Programme und der Binärfiles bemerkbar. In dem Anhang des Handbuchs werden die Möglichkeiten und Merkmale dieses schnellen DOS erklärt.

Fazit

Man erhält für etwa 180,- DM ein Textverarbeitungssystem, das für kleine bis mittlere Anwendungen kaum Wünsche offen läßt. Das 152 Seiten starke Handbuch erklärt gut verständlich die Benutzung der verschiedenen Programme. Eine mitgelieferte Referenz-Karte bietet einen schnellen Überblick über die zahlreichen Befehle. Die Programme werden auf einer beidseitig bespielten kopierbaren Diskette geliefert und laufen unter DOS 3.3 bzw. Diversi-DOS. Verblüffend ist, daß alle Programme in BASIC geschrieben sind. Verschiedene zeitintensive Routinen liegen jedoch als Maschinenunterprogramme vor. Zusätzlich enthält die Diskette noch einige Beispieltexte.

Die Ausführungsart von verschiedenen Befehlen ist gewöhnungsbedürftig. So wird das Einfügen für meine Begriffe etwas umständlich gehandhabt. Man muß erst für die gewünschte Anzahl von Zeilen Platz schaffen (Ctrl-I), kann dann Text eingeben und muß zum Schluß ggf. die beiden Textenden zusammenfügen (Ctrl-R). Der Gewöhnung bedarf auch das automatische Einfügen von zwei Leerzeichen im Anschluß an das Satzende nach dem Formatierbefehl (Ctrl-R). Die Zeit für das Löschen einzelner Zeichen wird etwas lang, da gleichzeitig alle Zeilen des Absatzes neu formatiert werden. Nach dem Löschen ganzer Wörter entstehen Lücken, die erst nach der Neuformatierung geschloßen werden.
Nach einiger Zeit Arbeit mit dem „Correspondent" komme ich jedoch zu der Ansicht, daß ich dieses Programm trotz der genannten Einschränkungen in Zukunft häufig benutzen werde.

Hard- und Software zum Apple IIc

zusammengestellt von Matthias Pohl

Die nachfolgenden Produkte, die speziell für den Apple IIc entwikkelt wurden, sind zum Teil noch nicht in der Bundesrepublik erhältlich.

Z80-Karte

Mit der CP/M-Karte von Precision Software erhält der Benutzer Zugang zur größten Standard-Software-Bibliothek der Welt. Zum Lieferumfang gehören neben der eigentlichen Karte, die den Z80-Prozessor beherbergt, das CP/M-Plus-Betriebssystem nebst Dokumentation. Die Hardware kann im Rechnergehäuse installiert werden; sie stört die anderen Betriebsarten nicht.

Maus und Joystick in einem

Kraft Systems versucht mit dem Quickstick IIc die Vorteile der Maus und des Joysticks zu vereinen: Mit einem Schieberegler kann zwischen beiden Betriebsarten geschaltet werden. Im Joystick-Modus kann man darüber hinaus wählen, ob der Steuerknüppel nach Loslassen von selbst in die Mittelstellung zurückkehren soll (Selbstzentrierung) oder ob er an der letzten Position verbleiben soll.

Drucker-Interface

Hotlink von Orange Micro verwandelt serielle in parallele Datenformate und ermöglicht so den Anschluß der meisten Drucker. Da die Chips auf dem Interface in CMOS-Technik ausgeführt sind, wird keine externe Spannungsversorgung benötigt. Ein Text/Graphics-Switch stellt die Kompatibilität mit den verschiedenen Drukker-Drivern sicher. Im Prinzip dieselbe Funktion wie Hotlink hat auch der Grappler C, allerdings ist er, im Unterschied zu ersterem, mit umfangreicher Firmware zur Ausgabe und Aufbereitung von Grafiken versehen: So lassen sich Hires-Bilder invers, gedreht oder in doppelter Größe ausdrucken.

Druckerpuffer

U-Print AP64 P hat einen 64K-fassenden, internen Speicher zur Pufferung von zur Druckausgabe anstehenden Daten. Mit einem Wählschalter kann die gewünschte Kopienanzahl eingestellt werden, ein zweiter Schalter dient der Speicherlöschung. Das Gerät ist mit einem Centronics-Interface ausgestattet. Unter der Bezeichnung U-Print AP16 ist auch ein kleinerer Typ mit 16K-Speicher erhältlich. Beide Geräte werden von Digital Devices hergestellt.

Noch ein Interface

Mit ApriCord IIc von Apricorn kann man centronics-kompatible Drukker an den Apple IIc anschließen. Zur Spannungsversorgung der in einem Kunststoff-Spritzguß-Gehäuse untergebrachten Elektronik ist kein eigenes Netzteil erforderlich.

Reisekoffer

Mit dem aus Polyethylen gefertigten und an den Seiten mit Alu-Profilen verstärkten Koffer von Fiberbilt ist ein sicherer Transport des Apple IIc möglich. Neben dem eigentlichen Rechner finden auch der Monitor und ein zweites Laufwerk Platz. Auch für Modem und Maus sind in dem Schaumstoffeinsatz Aussparungen vorhanden.

Paralleles Drucker-Interface

Mit der Serial Box von PBI lassen sich die Drucker von Epson, Okidata und C.Itoh an den Apple IIc anschließen. Alle zur Verbindung von Drucker und Computer benötigten Kabel werden mitgeliefert.

Transportkoffer

Der Transportkoffer von World Wide Case soll den Apple IIc bei Transport und auf Reisen schützen. Jeder Koffer ist innen ausgeschäumt und mit einem Schloß und Außentaschen versehen.

Buchhaltung

Das Buchhaltungsprogramm Rags to Riches von Chang Labs gibt es

jetzt auch für den Apple IIc. Es setzt sich aus den vier Modulen Hauptbuch, Außenstände, Zahlungen und Umsatz zusammen. Das Programm benötigt 128K und nutzt die Technik des Screen-Splitting.

Slimline-Laufwerk
Ad-3C ist ein 5,25-Zoll-Slimline-Laufwerk, das direkt an den Apple IIc angeschlossen werden kann. Es wird von American Mitac hergestellt und ist in der Farbgebung dem Rechner angepaßt.

Adapter
Mit Hilfe des AppleDaptor von Micro-Design kann man alle Disk Drives, die sich am Aplle II, II+ oder IIe betreiben lassen, auch an den Apple IIc anschließen.

5,25-Zoll-Laufwerk
Als zweites Laufwerk angeschlossen werden kann der Micro Drive von Titan Data Systems. Er ist in Slimline-Technologie ausgeführt und soll kürzere Zugriffszeiten haben als die Original-Apple-Laufwerke. Alle zum Anschluß benötigten Kabel sind Teil des Lieferumfangs.

Interface für Epson-Drucker
Unter Benutzung der Universal Card von Hanzon kann man die beiden Druckertypen Epson RX und Epson FX an den IIc anschließen. Das Interface ändert die für den Imagewriter ausgelegten Kontrollcodes dergestalt, das sie auch von den Epson-Druckern „verstanden" werden.

Grafikpaket
Dazzle Draw von Broderbund Software ist ein Programmpaket zur Grafikerstellung. Benötigt werden 128K-Speicher und die Double Hires-Grafik. Die Eingabe kann mittels Apple Mouse, Koala Pad, Grafik-Tablett oder Joystick erfolgen. (s. Peeker 9/85, S. 61.)

Synthesizer
Mockingboard von Sweet Micro Systems ist ein Synthesizer, der sechs Musikstimmen, Halleffekte und Sprache erzeugen kann. Zur Ausgabe der Toneffekte hat das in einem separaten Gehäuse untergebrachte Gerät Stereolautsprecher.

Kontrollsystem
Mit Smarthome von CyberLynx kann man sämtliche elektrischen Verbraucher zentral mit dem Apple IIc steuern und überwachen. Der Anschluß der Hardware erfolgt direkt an die serielle Schnittstelle des Rechners.

Lichtgriffel
Der Gibson Light Pen von Koala Technologies wird durch ein Kabel mit dem Rechner verbunden. Zum Lieferumfang gehören vier Programme, die die Möglichkeiten dieses Device ausschöpfen. Die Programme heißen Pen Painter, Pen Designer, Pen Animator und Pen Musician.

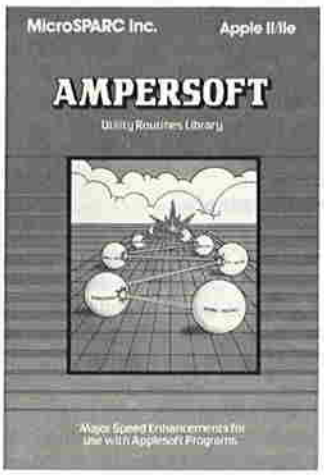
Spracherzeugung
Mit dem Cricket-Synthesizer von Street Electronics kann man sowohl menschlich klingende Stimmen (männliche und weibliche) als auch „Roboterstimmen" erzeugen. Hierzu hat das mit dem TI 5220 Sprachchip ausgestattete Gerät eigene Lautsprecher. Des weiteren gibt es sechs Musikkanäle, eine eingebaute Uhr und einen Kopfhöreranschluß.

Ampersoft

Eine Ampersand-Utility-Sammlung

Kurzbericht von Franz-Josef Hüskens

Ampersoft ist eine Utility-Sammlung, die von dem holländischen Statistik-Professor und Apple-Kenner C. Bongers erstellt wurde und über die Software-Abteilung der amerikanischen Zeitschrift „Nibble" vertrieben wird. Ein Teil der Utilities ist in der Nibble (z.B. die Sortierroutine) bzw. in „Call A.P.P.L.E." (z.B. der DOS-Mover) veröffentlicht worden.
Ampersoft liegt einschließlich des gemovten DOS 3.3 komplett in der Bank 1 und 2 der Language-Card. Mit Ausnahme der Matrizenrechnung sind alle anderen Routinen auch durch entsprechende Peeker-Beiträge behandelt worden, auf die gesondert hingewiesen wird.



DOS-Mover
Erfreulich ist, daß Ampersoft nicht nur Applesoft-BASIC erweitert, sondern zusätzlich das DOS 3.3 in die Language-Card schiebt und damit in den unteren 48K ca. 10000 Bytes mehr Platz schafft. Dieses verschobene DOS unterscheidet sich jedoch von dem normalen DOS 3.3. So gibt es kein INIT-Kommando mehr; man braucht also weiterhin das Standard-DOS, um Disketten initialisieren zu können. Für verschiedene Programme (COPYA, FID, MUFFIN, usw.) befinden sich auf der Diskette zusätzliche Routinen, die das jeweilige Programm an das veränderte DOS anpassen.
(Vgl. „DOS-Mover" in Peeker 2/86.)

CATALOG-Befehl
Nach einem CATALOG erhält jeder File auf der Diskette ein zusätzliches Zeichen wie -, =, / usw. als Abkürzung. Damit kann auf den gewünschten File zugegriffen werden, ohne den vollständigen Namen eingeben zu müssen.

Print-Using
Mit Ampersoft erhält Applesoft-BASIC ein schnelles Print-Using, das keine Wünsche mehr offen läßt. Man kann nun mit Hilfe von Spezial-Symbolen innerhalb einer Formatierungsmaske Zahlenkolonnen oder auch andere Informationen beliebig formatiert ausgeben. Die Ausgabe ist nicht nur über Bildschirm und Drucker, sondern auch auf die Diskette möglich.
(Vgl. „Print-Using" in Peeker 11/85.)

Matrizenbefehle
Die sehr leistungsfähigen Matrizenbefehle werden Gegenstand eines gesonderten Artikels im Peeker sein („Matrizenrechnung für Betriebswirte", voraussichtlich in Peeker 3/85).

Sortierroutine
Mit einer Sortierroutine können ein- und zweidimensionale Felder schnell in aufsteigender Folge sortiert werden. Dabei spielt es keine Rolle, ob es sich um Zahlen oder Strings handelt. Für mehrere, durch Indizierung miteinander verbundenen Feldern kann der Index entsprechend mitsortiert werden.
(Vgl. „Quicksort" in Peeker 1/86.)

BSAVE/BLOAD von Arrays
Zahlenfelder können mit Ampersoft als Binärdateien auf Diskette gespeichert werden. Dies bietet gegenüber der Ablage als Textfiles den Vorteil, daß die Zugriffszeit verkürzt und der auf der Diskette benötigte Speicherplatz verringert wird.
(Vgl. „ProDOS für Aufsteiger", Bd. 2, S. 87.)

Löschen von Arrays
Eine weitere Utility erlaubt das selektive Löschen von bereits dimensionierten Feldern. Damit kann in Applesoft-Programmen der vorhandene Speicherplatz effektiver ausgenutzt werden.
(Vgl. „Wie man Arrays löscht" in Peeker 2/84.)

Garbage-Collection
Die sehr langsame Garbage-Collection des Applesoft-Interpreters wird mit Ampersoft durch eine sehr schnelle ersetzt.
(Vgl. „Müllabfuhr wie ein Blitz" in Peeker 1/85.)

Alle Routinen können aus Applesoft-Programmen heraus mit einfachen &-Befehlen aufgerufen werden. Aus diesem Grunde funktionieren die Utilities nicht bei compilierten Applesoft-Programmen.

Ampersoft wird auf einer kopierbaren Diskette geliefert. Im zugehörigen Handbuch wird die Benutzung der Utility-Sammlung anhand von Beispielen eingehend beschrieben und erklärt. Die einzelnen Utilities erweitern das etwas karge Applesoft-BASIC zwar um einige wichtige Befehle, jedoch ist der Preis von etwa 200,- DM für meine Begriffe überhöht. Es gibt Programme, die gleiches oder ähnliches leisten, aber erheblich preiswerter sind. Der Vorteil liegt in der Bündelung, falls man alle Ampersand-Routinen gleichzeitig benötigt, weil man sich sonst einzelne Routinen (aus dem Peeker usw.) erst zusammenstellen müßte.

Buchbesprechungen

Was ist wo im Apple
1985, 496 S., zahlr. Tabellen und Abb., kart.
von Wiliam F. Luebbert
Sybex-Verlag, Düsseldorf
Dieses Buch ist eine gelungene Übersetzung des schon fast legendären „What's Where in the Apple", das in den Anfängen des Apple eine der meistzitierten Quellen war und nun auch für den IIe auf den neusten Stand gebracht wurde.
Es bietet sowohl für Assembler- als auch für gehobene Applesoft-Programmierer eine Fülle von Tips und Informationen, von der Belegung der Zero-Page über die Organisation des Bildschirms bis hin zur Beschreibung von DOS, Applesoft-Interpreter und Monitor.
Der numerische und alphabetische Atlas (über 140 Seiten) faßt alle Speicherstellen und Routinen zusammen und beschreibt kurz deren Anwendung. Die Fülle an Informationen über DOS, beide BASIC-Interpreter und II+/IIe-Monitor ermöglicht eine effiziente Ausnutzung des Apple, wobei die hier angegebenen Label-Namen quasi zum Standard wurden.
Das Buch bietet selbst dem erfahrenen Apple-Programmierer noch neue Möglichkeiten und kann auch später im täglichen Gebrauch als nützliches Nachschlagewerk dienen.
Inhalt
Systemspezifische Programmierung – PEEK, POKE und CALL – Apple-Architektur – Adressierung beim 6502 – Maschinenprogramme in BASIC-Umgebung – Die Speicherorganisation – Monitor- und DOS-Vektoren – Bildschirmspeicher – Benutzerspeicher – Das Diskettenbetriebssystem DOS – Der E/A-Bereich – Der Applesoft-BASIC-Interpreter – Der Monitor-Bereich – Atlas

Apple-Assembler lernen
Bd. 1, Einführung in die Assembler-Programmierung des 6502/65C02
von Jürgen Kehrel
1985, 234 S., zahlr. Abb., kart., DM 38,-
Hüthig Verlag, Heidelberg
Dieses Buch bietet in 35 Lektionen einen bequemen Einstieg in die Assembler-Programmierung auf dem Apple. Ausgehend von Applesoft-BASIC werden kleine Maschinenprogramme entwickelt, wobei besonderer Wert auf eine didaktische Einführung in die Programmierung mit einem Assembler gelegt wurde.
In jedem Fall empfehlenswert ist der Kauf der Begleitdiskette (Preis: DM 44,-), die neben allen in dem Buch gezeigten Programmen auch einen „ausgewachsenen" Assembler mit Editor bietet, der trotz der fehlenden Makro-Eigenschaft durchaus mit professionellen Assemblern konkurrieren kann und weit über die Anfänge der Programmierung in Maschinensprache alle Wünsche abdeckt. Ein besonders für Anfänger fast unausweichliches Hilfsmittel ist der ebenfalls auf der Begleitdiskette enthaltene Debugger IDUS, der es ermöglicht, die selbstgeschriebenen Programme schrittweise zu verfolgen und die Änderung von Speicherstellen, Registern und Flaggen zu beobachten.
Vor der Besprechung der Befehle erfolgt eine leichtverständliche Einführung in die Welt der Bits und Bytes. Bei den Programmen wurde versucht, Probleme der täglichen Assembler-Praxis aufzugreifen, so daß nicht unnütze Probleme besprochen werden, sondern schon einmal eine verständliche aber professionelle Routine angeboten wird.
Die für Anfänger zunächst undurchschaubaren Adressierungsarten des 6502 werden nicht nebenbei beschrieben, sondern am Ende der Einführung mit instruktiven Abbildungen veranschaulicht, ein Weg, den jeder Assembler-Kurs beschreiten sollte. Eine umfangreiche Zusammenfassung des kompletten Befehlssatzes mit Beispielen für den täglichen Gebrauch und verschiedene Tabellen runden den ersten Band ab.
Inhalt
Assembler-Kurs in 35 Lektionen – Der 6502/65C02 Befehlssatz – Der Assembler ASSESSOR – Der Debugger und Simulator IDUS – Anhang

ProDOS-Analyse
Versionen 1.0.1, 1.0.2, 1.1.1
von Arne Schäpers
1985, 470 S., kart., DM 68,-
Hüthig Verlag, Heidelberg
Mit der ProDOS-Analyse erhält der Leser eine umfassende Darstellung der kompletten ProDOS-Interna, ohne das BASIC.SYSTEM. Da sowohl die Versionen 1.0.1 als auch die geänderten Versionen 1.0.2 und 1.1.1 beschrieben werden, kann der mit Assembler vertraute Anwender (und nur für diese Gruppe ist dieses Buch geeignet) sein Betriebssystem modifizieren und patchen oder auch nur ergründen. Die kommentierten Assemblerlistings aller Teile (MLI mit RWTB, Boot-Programm, RAM-Disk etc.) geben einen tiefgreifenden Einblick in die Arbeitsweise des Betriebssystems. Dabei wird jeweils vorher die genaue Funktion der einzelnen Komponenten beschrieben, so daß nach der Lektüre dieses Buches kaum noch Fragen zu dem Betriebssystem offen bleiben dürften.
Mit der ProDOS-Analyse hat der Autor selbst das Technical Reference Manual der Firma Apple übertroffen und die wohl umfangreichste theoretische Darstellung eines Betriebssystems geschaffen, die je auf dem freien Markt erhältlich war.
Das MLI – Dateistrukturen unter ProDOS – Programmiertricks im MLI – BOOT und REBOOT von ProDOS – Die Verarbeitung von NMI, RESET und IRQ – SET TIME für die Thunderclock – Die System Global Page – Die RWTB der DISK II – Die Organisation einer Diskette – Hardware Details – Die RAMDisk – Updates von ProDOS – Anhänge

Bewegte Apple-Grafik
DOS Toolkit-Erweiterungen
von Arne Schäpers
1985, 308 S., kart., DM 58,-
Hüthig Verlag, Heidelberg
Die DOS Toolkit-Diskette enthält neben dem Assembler auch ein interessantes Programm zur Darstellung von Objekten auf dem HGR-Bildschirm des Apple. Dieser Hires Character Generator ist Gegenstand dieses Buches, das den fortgeschrittenen Applesoft-Programmierer mit Assemblerkenntnissen in die Welt der bewegten Grafik einführt. Im ersten Teil des Buches wird zunächst der HRCG analysiert und dessen Möglichkeiten aufgezeigt. Daran schließt sich ein kommentiertes Assemblerlisting an.
Im zweiten Teil wird ein professionelles Arkade-Spiel entwickelt, bei dem der Leser unter anderem lernt, wie simultane Bewegungen mehrerer Objekte gleichzeitig realisierbar sind. Dabei wird auch auf die Steuerung mit Joysticks und Paddles eingegangen. Dieser Blick hinter die Kulissen der Programmierung von Computerspielen bietet die Möglichkeit, eigene Ideen umzusetzen und Spiele zu programmieren, die in nichts den professionellen Angeboten nachstehen müssen.
Inhalt
Der HRCG – Line-Skewing – Transparente Overlays – Arkade-Algorithmen – Asymmetrische Figuren und optische Mittelpunkte – Kontrollprogramm – Kollisionen – Ausblicke

Applesoft BASIC
Tips und Tricks
von Frank Bühler
1985, 244 S., kart., DM 38.-
Hüthig Verlag, Heidelberg
Dieses Buch ist für den Anfänger und Fortgeschrittenen gleichermaßen nutzbar. Neben einer Einführung in die Programmiersprache BASIC, die für den Einstieg aus einer anderen Sprache ausreichend sein dürfte, werden alle Applesoft-Befehle mit entsprechenden Beispielen aufgeführt.
Darüber hinaus bietet das Buch einige Programmbeispiele mit z.T. kleineren Assembler-Routinen (deren Eingabe in den Rechner leider nicht erklärt wird), die dann in eigene Programme aufgenommen werden können. Am Ende folgen einige Tips zur Entwicklung von effizienten Applesoft-Programmen. Im Anhang findet der Leser noch einmal Tabellen mit Fehlermeldungen, 6502-Befehlssatz und nützliche Adressen.
Inhalt
Applesoft-Basic im Vergleich – Grundelemente von Basic – Applesoft-Befehlssatz – Rechenoperationen – Anwendungs- und Programmbeispiele – Programmentwicklung/Hinweise – Anhang

dBase II
Bd. 2: Programmierung
von W. Eggerichs
1985, 191 S., zahlr. Abb., kart., DM 38,90
Hüthig Verlag, Heidelberg
Das Buch knüpft unmittelbar an den Band 1 „Einführung" an. Mit seiner kurz-sachlichen Formulierung hebt es sich wohltuend von zahlreichen dBase-Büchern ab, die weitschweifend wenig Information pro Seite bieten.
An den kleinen Beispielen werden die grundlegenden Befehle zum Programmieren in dBase erklärt. Zur grafischen Darstellung wird das Struktogramm gewählt (auf den wesentlich unübersichtlicheren PAP verzichtet der Autor zu Recht). Abschließend wird eine vollständige Anwendung zur Verwaltung von Kundenstammdaten wiedergegeben, die der Leser leicht seinen eigenen Wünschen anpassen kann.

In Vorbereitung

Aufsätze

Dos-Mover
Erhöhung der Speicherkapazität um 30%
(DOS 3.3)

Binäres Rechnen mit Papier und Bleistift
Teil 1: Addition und Subtraktion
(Grundlagen)

Matrizenrechnung in der betriebswirtschaftlichen Praxis
(Praxis)

Der IEC-Bus
(Hardware)

Serielle Schnittstellen beim Apple IIc
(Hardware)

MAKESUB
Erstellung zusammenhängender Subdirectories
(ProDOS)

Befehlserweiterung durch CHRGET-Manipulation
(Applesoft)

Bildschirmmasken in der Language-Card
(Assembler)

Dreidimensionale Funktionsdarstellung
(Grafik)

Dreiecksberechnung
(UCSD)

ReadIn von Integer-Zahlen
(UCSD)

Test- und Erfahrungsberichte

CP/M-3.0-Karte für Apple IIc
(Semjan)

Premium-Softcard
Speicherverwaltung unter CP/M
(Microsoft)

FSS 280
Diskettenlaufwerk
(erphi electronic)

Prometric
65C02C- und Z80B-Rechner
(Electronic Special Service)

Microbuffer II
Druckerpuffer-Karte
(Practical Peripherals)

ML-192-Drucker
(Okidata)

SG-15-Drucker
(Star)

Mockingboard
Musik- und Sprachkarte
(Sweet Micro Systems)

Multi-Disk-Catalog III
Diskettenverwaltung
(Sensible Software)

DCODE
Applesoft-Hilfsprogramme
(Beagle Brothers)

Sammeldisk #13

QUICK.RANDOM (Heft 1/86, S. 6ff.)
QUICK.SPEZIAL
QUICK.TASC
QUICK.DISK

QUICKSORT.DEMO (Heft 1/86, S. 16ff.)
T.QUICKSORT
QUICKSORT

VOK.TRAINER (Heft 1/86, S. 20ff.)
VOK.COPY
VOK.PACK
VOK.BCOPY
GWS.INFO
GWS.VOK

AGE.DEMO (Heft 1/86, S. 30ff.)
T.AGE
AGE

GRAFIK.DEMOS.2 (Heft 1/86, S. 61ff.)

Achtung!

Wegen der riesigen deutsch-englischen Übungsdatei GWS.VOK zum Vokabeltrainer müssen die nachfolgenden Programme aus Heft 1/86 auf die Sammeldisk #14 übernommen werden, die im übrigen die Programme aus Heft 2/86 enthalten wird.

T.PROTODOS (Heft 1/86, S. 36ff.)
PROTODOS

DESIGNER.TEXT (Heft 1/86, S. 43ff.)

READPAS.PAS (Heft 1/86, S. 48ff.)

Apple- und Peeker-Händler

Den „Peeker" gibt es bei folgenden Apple-Händlern:

Beso GmbH, Oswaldstr. 5, 4700 Hamm 4
Böhmer Electronic, Wilhelm-Zopf-Str. 9, 7080 Aalen
Büro Plan Klaus Bayerl, Bochumer Str. 22, 4270 Dorsten
City Computer, Dönhoffstr. 23, 5090 Leverkusen
Computerland, Am Wall 137, 2800 Bremen 1
Delta-Electronic, Gugelmattstr. 31, Ch-8967 Widen
DMS Büroorganisation GmbH, Schwimmbadstr. 36, 6301 Heuchelheim
Günther Heinicke GmbH, Postfach 80 04 26, 2050 Hamburg 80
Horten Warenhaus, 1 N 7, 2a, 6800 Mannheim
Norbert Hunstig, Nottulner Landweg 81, 4400 Münster
Karstadt AG, Kampstr. 1, 4600 Dortmund 1
Kaut-Bullinger & Co. GmbH, Rosenstr. 8, 8000 München 8
Josef Kraus, Darmstädter Str. 26, 6148 Heppenheim
Microland-Computer GmbH, Bäringer Str. 31, 3380 Goslar
Mittelrheinisches Rechenzentrum GmbH, Postf. 128, 5160 Düren
Novocomp Datensysteme GmbH, Walramsneustr. 7 u. 9, 5500 Trier
Orgasoft GmbH, Graneggstr. 43, 7732 Niedereschach
Pandassoft, Uhlandstr. 195, 1000 Berlin 12
R + R Electronic, Breslauerstr. 29, 6900 Heidelberg 1
Schöpp GmbH, Bahnstr. 79, 6140 Bensheim
Staehlin-Büroorganisation, Klostersteige 12-14, 8960 Kempten / Allgäu
Wagner Datentechnik, Hochstr. 1, 7990 Friedrichshafen
Würth Bürobedarf + Organisation GmbH, Am Rupertsberg 16, 6530 Bingen 1

Inserentenverzeichnis Peeker 1/86

aaa electronic gmbh, Freiburg 35
ACS, Detmold 15
AFC Computer GmbH, Köln 23
Ampersand (vorm. Pandabooks), Berlin 41
ccp-datentechnik, Hamburg 23
Ingenieurbüro Fricke, Berlin 13
GEPARD Computer GmbH, Oldenburg 23
HIB Computerladen, Nürnberg 14
N. Hunstig, Münster 15
Interkom electronic, Isernhagen 15
Intus, Waldshut-Tiengen 23
Jeschke, Kelkheim 23
U. Mohwinkel Electronic, Leverkusen 13
Pandasoft, Berlin 53
M. Semjan Computer Systeme, Frankfurt 15
Tewi-Verlag, München . . . . . 2. US
Ueding electronics, Menden 39

Hüthig-FACHBUCH-TIP

„Apple Assembler" wendet sich an alle, die bereits Anfängerkenntnisse der 6502-Programmierung haben und nunmehr ein Nachschlagewerk für ihren Apple II Plus/II e/II c suchen, in dem alle wichtigen ROM-Routinen sowie eine Vielzahl sonstiger Hilfsprogramme in einer systematischen Form zusammengestellt werden. Insgesamt umfaßt dieses Buch über 40 Utilities, darunter mehrere völlig neuartige Programme wie Double-Lores, Double-Hires, Screen-Format u. a.

Der erste Teil enthält ein Repetitorium der wichtigsten Befehle, Adressierungsarten und sonstigen Besonderheiten des 6502 sowie Angaben zu den apple-spezifischen Zahlenformaten (Integer- und Fließkommazahlen). Im zweiten Teil werden neben einer Kurzwiederholung der Monitor-Befehle alle Routinen und Adressen des Monitors zusammengestellt, die für Assemblerprogrammierer von Nutzen sein können. Darüber hinaus findet der Leser Unterroutinen für Vorwärts- und Rückwärts-Moven, hexadezimale Addition/Subtraktion/Multiplikation/Division, Binär-, Hex-und ASCII-Umwandlung, Dumpen/Disassemblieren, Aufwärts-Scrollen, Reset u. a. Der dritte Teil befaßt sich mit der Speicherverwaltung der Language-Card und der II e-64K-Karte und enthält Test- und Move-Programme zum Verschieben von Daten in die und aus der Language Card sowie der 64K-Karte, wobei besonders ausführlich auf die Softswitches eingegangen wird.

Der vierte Teil ist dem Applesoft-ROM gewidmet und beschreibt die interne Struktur von Applesoft-Programmen, die Methoden der Parameterübergabe mittels CALL, USR, &, PEEK und POKE und listet dann eine große Anzahl nützlicher Interpreter-Adressen. Bei den Utility-Programm liegt das Schwergewicht auf Fließkommamathematik einschließlich Print Using.

Der letzte Teil behandelt den Text- und Grafikspeicher. Neben einem professionellen Maskengeneratorprogramm findet der Leser hier auch erstmals Routinen zur Double-Lores- und Double-Hires-Grafik des Apple II e.

Apple Assembler - Tips und Tricks -

von U. Stiehl

232 S., 40 Programm-Listings, 3 Abb., kart., DM 34,—
ISBN 3-7785-1047-9

Begleitdiskette zum Buch DM 28,—
ISBN 3-7785-1048-7

BESTELLCOUPON

Bitte ausfüllen und an Hüthig Vertriebsservice, Postfach 102869 · 6900 Heidelberg schicken.

Buchtitel

Name

Straße

Ort

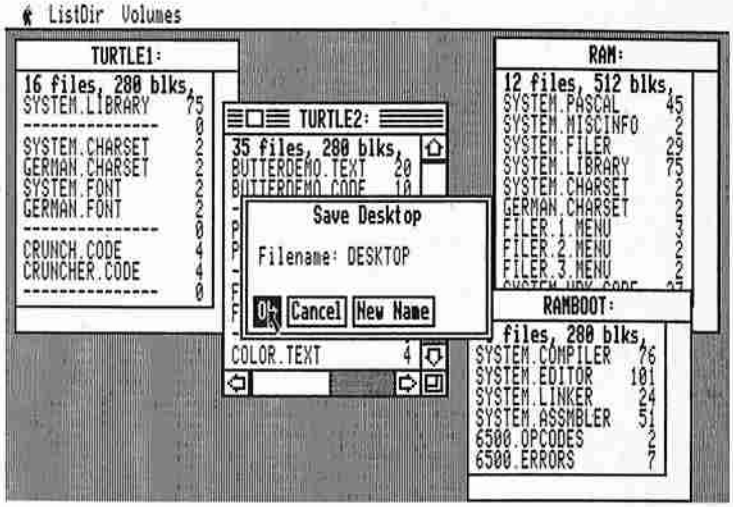
Unterschrift

In Vorbereitung

TurtleGraphics-Library-Paket

von Dieter Geiß

Turtle-Utilities für Fenstertechnik und Apple-Maus in einfacher und doppelter Hires-Grafik für Pascal 1.1/1.2 auf Apple II+/e/c mit Maus oder Joystick.
2 Disketten mit umfangreichem Manual, DM 98,–
Erscheinungstermin Anfang 1986



Das Utility-Paket besteht aus vier Modulen, die von Programmierern benutzt werden können, um professionelle grafische Anwendungsprogramme in Pascal zu schreiben.

Benötigt wird ein Apple Pascal Betriebssystem, entweder die Version 1.1 oder die neue Version 1.2. Bestehende Programme laufen ohne Einschränkung mit der neuen „TurtleGraphics", wenn diese nicht zu viel Speicherplatz verbraucht haben, da die neue „TurtleGraphics" umfangreicher als die alte ist.

Im einzelnen bietet das Paket folgende Möglichkeiten:

- volle Kompatibilität mit der alten „TurtleGraphics"
- lauffähig auf Pascal 1.1 und 1.2
- funktionsfähig mit angeschlossener UltraTerm-Karte
- alle zeitkritischen Funktionen in reinem Assembler programmiert
- Benutzung der zweiten Hires-Seite ($4000–$5FFF) möglich
- für Apple IIc und Apple IIe mit erweiterter 80-Zeichen-Karte Benutzung der doppelten Hires-Grafik mit 560 × 192 Punkten bzw. 16 neuen Farben möglich
- schnelle Prozeduren zum Zeichnen eines Punktes oder einer Linie
- Linearisierung von Teilen des Hires-Schirms
- Benutzung mehrerer Zeichensätze gleichzeitig
- Laden und Speichern von Hires-Bildern mit Ausdruck über Pascal-SUPERDUMP
- Scrolling des Hires-Schirms oder eines Teils in vier Richtungen
- drei verschiedene Schriftarten: Fett-, Breit- und Proportionalschrift, beliebig mischbar (acht Möglichkeiten)
- spezielle schnelle Ausgabe von Text
- Cursor bei Eingabe frei programmierbar
- Ein-/Ausgabe von INTEGER-, CHAR-, STRING- und REAL-Werten im Grafikmodus
- Menüzeile wie beim Macintosh (Die nachfolgenden Module benötigen Maus/Joystick)
- Pull-down-Menüs
- Laden und Speichern von Fenstern (Windows)
- Öffnen von Fenstern
- Aktivieren und Deaktivieren von Fenstern
- Verschieben und Vergrößern/Verkleinern von Fenstern
- Scrolling von Fensterinhalten in allen vier Richtungen
- Umfangreiche Demos als Quelltexte.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg